# YoctoHub-Ethernet, Mode d'emploi

# Table des matières


**1. Introduction** ..... **1**

**2. Présentation** ..... **3**

2.1. Les éléments du YoctoHub-Ethernet ..... 3

**3. Premiers pas** ..... **7**

3.1. Utilisation directe pour les impatients ..... 7
3.2. Configuration manuelle ..... 8
3.3. Configuration automatisée ..... 10
3.4. Connexions ..... 11

**4. Montage** ..... **15**

4.1. Fixation ..... 15
4.2. Fixation d'un sous-module ..... 16

**5. Utilisation du YoctoHub-Ethernet** ..... **17**

5.1. Localisation des modules ..... 17
5.2. Test des modules ..... 18
5.3. Configuration des modules ..... 18
5.4. Upgrades des firmwares ..... 19

**6. Contrôle d'accès** ..... **21**

6.1. Accès "admin" protégé ..... 22
6.2. Accès "user" protégé ..... 22
6.3. Influence sur les API ..... 22
6.4. Effacement des mots de passe ..... 22

**7. Interactions avec l'extérieur** ..... **23**

7.1. Configuration ..... 23
7.2. User defined callback ..... 24
7.3. Yocto-API callback ..... 26
7.5. Thinkspeak ..... 26

**8. Programmation** ..... **27**

8.1. Accès aux modules connectés ..... 27

8.2. Contrôle du YoctoHub-Ethernet ..... 27

**9. Personnalisation de l'interface Web ..... 29**

9.1. Utilisation ..... 29
9.2. Limitations ..... 30

**10. Référence de l'API de haut niveau ..... 31**

10.1. Interface d'un port de Yocto-hub ..... 32
10.2. Interface de la fonction Network ..... 61
10.3. Interface de la fonction Files ..... 122

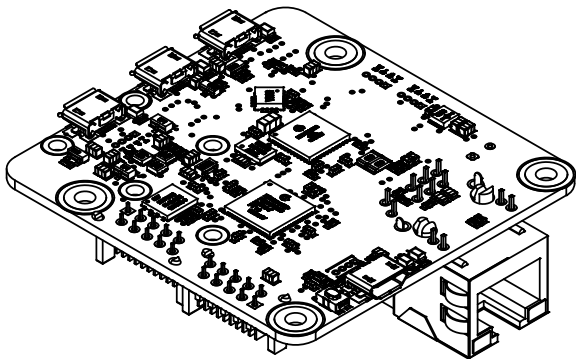
**11. Caractéristiques ..... 155**

Blueprint ..... 157

**Index ..... 159**

# 1. Introduction

Le module YoctoHub-Ethernet est un module de 60x58mm qui permet de contrôler d'autres modules Yoctopuce à travers Ethernet. Vu de l'extérieur, ce module se comporte exactement comme un ordinateur classique faisant tourner un VirtualHub[1]: même interface, mêmes fonctionnalités. Le YoctoHub-Ethernet peut être alimenté classiquement à l'aide d'un chargeur USB, mais il peut aussi être aussi alimenté par *Power over Ethernet*.



*Le YoctoHub-Ethernet*

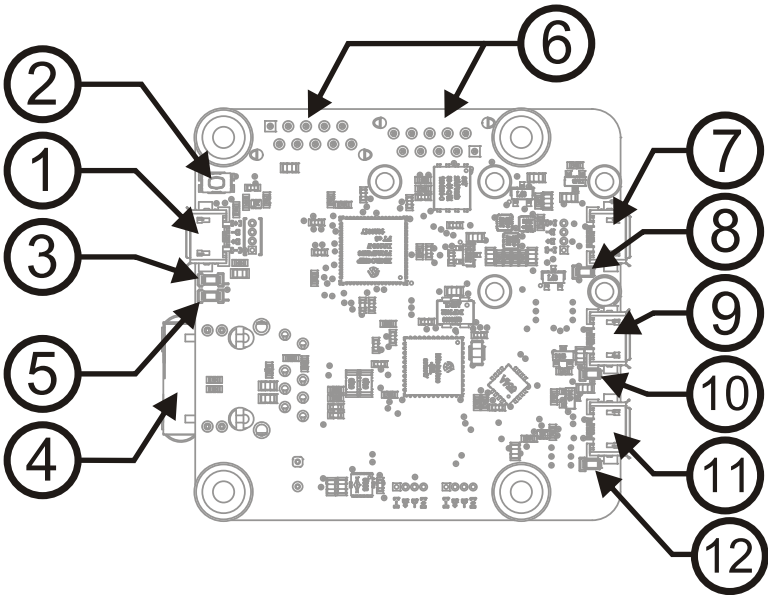
Le YoctoHub-Ethernet a été conçu pour être déployé facilement et ne pas demander de maintenance particulière. Contrairement à un mini-PC, il n'utilise pas un système d'exploitation complexe. Quelques réglages simples permettent son utilisation dans toutes sortes d'environnements réseau. Ces réglages peuvent être effectués manuellement ou de manière automatisée, par USB. Il convient de ce fait beaucoup mieux à une industrialisation qu'un mini-PC. En revanche, il ne permet pas l'exécution de programmes supplémentaires écrits par l'utilisateur.

Le YoctoHub-Ethernet n'est pas un hub USB standard avec accès réseau. Bien qu'utilisant du câblage USB, ses ports descendants utilisent un protocole propriétaire, plus simple qu'USB. Il n'est par conséquent pas possible de contrôler, ni même d'alimenter, des périphériques USB standards avec un YoctoHub-Ethernet.

Yoctopuce vous remercie d'avoir fait l'acquisition de ce YoctoHub-Ethernet et espère sincèrement qu'il vous donnera entière satisfaction. Les ingénieurs Yoctopuce se sont donnés beaucoup de mal pour que votre YoctoHub-Ethernet soit facile à installer n'importe où et soit facile utiliser en toutes circonstances. Néanmoins, si ce module venait à vous décevoir, n'hésitez pas à contacter le support Yoctopuce[2].

[1] http://www.yoctopuce.com/FR/virtualhub.php
[2] support@yoctopuce.com

# 2. Présentation




*1: Port USB de contrôle + alimentation*
*2: Yocto-bouton*
*3: Yocto-Led*
*4: Port réseau (100Mb)*
*5: Indicateur de sur-consommation*
*6: Connection dorsale*
*7: Port descendant 1*
*8: Indicateur port descendant 1*
*9: Port descendant 2*
*10: Indicateur port descendant 2*
*11: >Port descendant 3*
*12: Indicateur port descendant 3*

## 2.1. Les éléments du YoctoHub-Ethernet

### Le numéro de série

Chaque Yocto-module a un numéro de série unique attribué en usine, pour les modules YoctoHub-Ethernet ce numéro commence par YHUBETH1. Le module peut être piloté par logiciel en utilisant ce numéro de série. Ce numéro de série ne peut pas être changé.

### Le nom logique

Le nom logique est similaire au numéro de série, c'est une chaîne de caractères sensée être unique qui permet référencer le module par logiciel. Cependant, contrairement au numéro de série, le nom logique peut être modifié à volonté. L'intérêt est de pouvoir fabriquer plusieurs exemplaires du même

projet sans avoir à modifier le logiciel de pilotage. Il suffit de programmer les mêmes noms logiques dans chaque exemplaire. Attention, le comportement d'un projet devient imprévisible s'il contient plusieurs modules avec le même nom logique et que le logiciel de pilotage essaye d'accéder à l'un de ces modules à l'aide de son nom logique. A leur sortie d'usine, les modules n'ont pas de nom logique assigné, c'est à vous de le définir.

## Le Yocto-bouton

Le Yocto-bouton a deux fonctions. Premièrement, il permet d'activer la Yocto-balise (voir la Yocto-Led ci-dessous). Deuxièmement, si vous branchez un Yocto-module en maintenant ce bouton appuyé, il vous sera possible de reprogrammer son firmware avec une nouvelle version. Notez qu'il existe une méthode plus simple pour mettre à jour le firmware depuis l'interface utilisateur, mais cette méthode-là peut fonctionner même lorsque le firmware chargé sur le module est incomplet ou corrompu.

## La Yocto-Led

En temps normal, la Yocto-Led sert à indiquer le bon fonctionnement du module: elle émet alors une faible lumière bleue qui varie lentement mimant ainsi une respiration. La Yocto-Led cesse de respirer lorsque le module ne communique plus, par exemple s'il est alimenté par un hub sans connexion avec un ordinateur allumé.

Lorsque vous appuyez sur le Yocto-bouton, la Led passe en mode Yocto-balise: elle se met alors à flasher plus vite et beaucoup plus fort, dans le but de permettre une localisation facile d'un module lorsqu'on en a plusieurs identiques. Il est en effet possible de déclencher la Yocto-balise par logiciel, tout comme il est possible de détecter par logiciel une Yocto-balise allumée.

La Yocto-Led a une troisième fonctionnalité moins plaisante: lorsque le logiciel interne qui contrôle le module rencontre une erreur fatale, elle se met à flasher SOS en morse[1]. Dans ce cas, débranchez puis re-branchez le module. Si le problème venait à se reproduire, vérifiez que le module contient bien la dernière version du firmware et, dans l'affirmative, contactez le support Yoctopuce[2].

## Le connecteur de contrôle et d'alimentation (Power / Control port)

Ce connecteur permet d'alimenter le YoctoHub-Ethernet et les modules qui lui sont connectés à l'aide d'un simple chargeur USB. Ce connecteur permet aussi de prendre le contrôle du YoctoHub-Ethernet par USB, exactement comme on pourrait le faire avec un module Yoctopuce classique. C'est particulièrement utile lorsque que l'on désire configurer le YoctoHub-Ethernet sans connaître son adresse IP.

## Les ports descendants

Vous pouvez connecter jusqu'à trois modules Yoctopuce sur ces ports. Ils seront alors accessibles comme s'ils étaient branchés à un ordinateur faisant tourner un VirtualHub. Attention, le protocole entre le YoctoHub-Ethernet et le module Yoctopuce n'est pas de l'USB mais un protocole propriétaire plus léger. De ce fait le YoctoHub-Ethernet ne peut pas gérer des périphériques autres que des modules Yoctopuce. Un hub USB standard ne fonctionnera pas non plus[3]. Si vous désirez brancher plus de trois modules Yoctopuce, utilisez le connecteur dorsal[4].

Attention, les connecteurs USB du YoctoHub-Ethernet sont simplement soudés en surface et peuvent être arrachés si la prise USB venait à faire fortement levier. Si les pistes sont restées en place, le connecteur peut être ressoudé à l'aide d'un bon fer et de flux. Alternativement, vous pouvez souder un fil USB directement dans les trous espacés de 1.27mm prévus à cet effet, près du connecteur.

[1] court-court-court long-long-long court-court-court
[2] support@yoctopuce.com
[3] Le Micro-USB-Hub fabriqué par Yoctopuce est un hub USB standard et ne fonctionnera pas avec le YoctoHub-Ethernet.
[4] Vous trouverez très prochainement sur le site de Yoctopuce un produit YoctoHub-Shield, qui peut être chaîné au connecteur dorsal du YoctoHub-Ethernet pour y brancher quatre modules supplémentaires.

## Le connecteur UTP (réseau)

Ce connecteur permet de connecter le YoctoHub-Ethernet à votre réseau Ethernet. La connection réseau du YoctoHub-Ethernet fonctionne en 100 Mb/s, mais le hub peut très bien être raccordé à un réseau Gigabit ou 10Mb/s.

Le YoctoHub-Ethernet peut aussi être alimenté par ce biais: il suffit de disposer de matériel réseau capable de fournir de l'énergie par *Power over Ethernet* (PoE) à la norme 802.3af. Ce peut être un switch réseau avec PoE, ou un simple injecteur par exemple. Longtemps réservé au monde professionnel, ce type de matériel est désormais largement distribué à des prix tout à fait abordables. En utilisant le PoE, vous pourrez déporter votre YoctoHub-Ethernet loin de toute prise électrique, en le raccordant uniquement par un simple câble réseau. Les câbles réseau plats et fins sont généralement compatibles avec le standard PoE 802.3af.

## Les sondes de courant

Le YoctoHub-Ethernet est capable de mesurer sa propre consommation de courant, en distinguant celle fournie par le connecteur USB et celle fournie par le Power over Ethernet. La distribution du courant sur un bus USB ou sur un réseau PoE étant relativement critique, cette fonctionnalité peut être d'un grand secours. Le YoctoHub-Ethernet est dimensionné pour pouvoir gérer jusqu'à 2A en tout (propre consommation et consommation sur les ports descendants incluses).

## Indicateur de sur-consommation

Le YoctoHub-Ethernet analyse en permanence sa consommation. S'il détecte une consommation globale de plus de 2A ou une consommation supérieure à 1.8A sur le PoE, suite à une surcharge sur un des ports descendants par exemple, il va automatiquement désactiver tous les ports descendants et allumer la Led l'indicateur de sur-consommation. Pour isoler la source du problème, vous pouvez réactiver les ports un à un, en surveillant l'augmentation de la consommation. Alternativement, si connaissez la source du problème de sur-consommation et savez l'avoir résolu, vous pouvez redémarrer le YoctoHub-Ethernet pour réactiver tous les ports.

Notez que l'indicateur de sur-consommation est une mesure de protection qui peut éviter la surchauffe, mais ce n'est pas une garantie de protection contre les court- circuits.

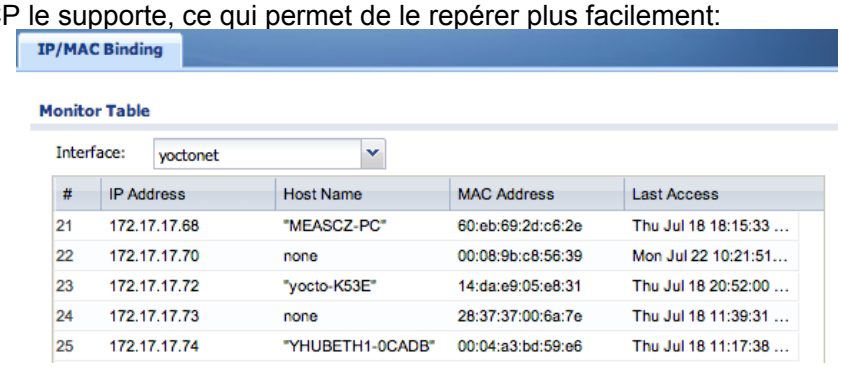
# 3. Premiers pas

Ce chapitre a pour but de vous aider à connecter et configurer votre YoctoHub-Ethernet pour la première fois

## 3.1. Utilisation directe pour les impatients

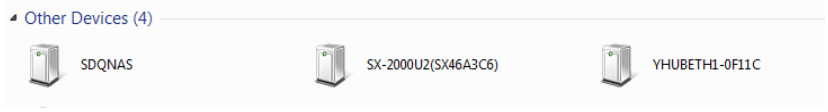
Si vous branchez votre YoctoHub-Ethernet directement au réseau sans le configurer, il est fonctionnel avec la configuration par défaut:

- L'adresse IP est obtenue par DHCP. Si aucun serveur DHCP ne répond, il utilisera une adresse d'auto-configuration dans la plage 169.254.x.y, comme la plupart des systèmes d'exploitation.
- Le YoctoHub-Ethernet indique au serveur DHCP son nom d'hôte (numéro de série), si le serveur DHCP le supporte, ce qui permet de le repérer plus facilement:


IP/MAC Binding
Monitor Table
Interface:
yoctonet
# IP Address Host Name MAC Address Last Access
21 172.17.17.68 "MEASCZ-PC" 60:eb:69:2d:c6:2e Thu Jul 18 18:15:33 ...
22 172.17.17.70 none 00:08:9b:c8:56:39 Mon Jul 22 10:21:51...
23 172.17.17.72 "yocto-K53E" 14:da:e9:05:e8:31 Thu Jul 18 20:52:00 ...
24 172.17.17.73 none 28:37:37:00:6a:7e Thu Jul 18 11:39:31 ...
25 172.17.17.74 "YHUBETH1-0CADB" 00:04:a3:bd:59:e6 Thu Jul 18 11:17:38 ...

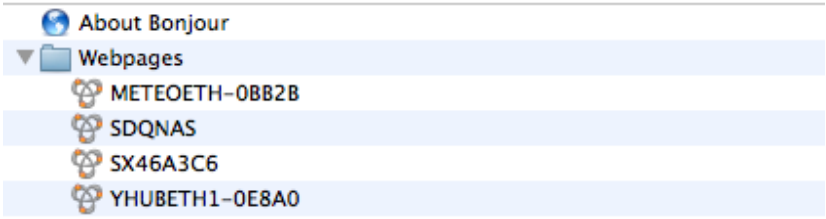
*YoctoHub-Ethernet détecté par le serveur DHCP d'un petit routeur ZyXEL*

- Le YoctoHub-Ethernet annonce sa présence sur le réseau local par le protocole *SSDP (uPNP)*, ce qui le rend visible par les machines Windows:




*YoctoHub-Ethernet détecté sur le réseau par un PC sous Windows*

- Le YoctoHub-Ethernet annonce sa présence sur le réseau local par le protocole *Bonjour*, ce qui le rend visible par les machines Mac OS X:

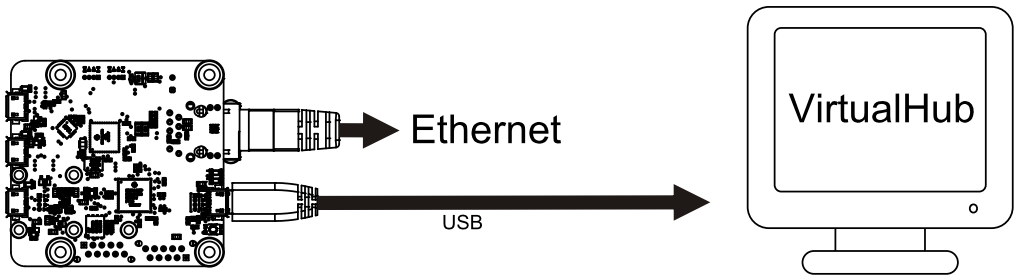



*YoctoHub-Ethernet détecté sur le réseau par un Mac*

## 3.2. Configuration manuelle

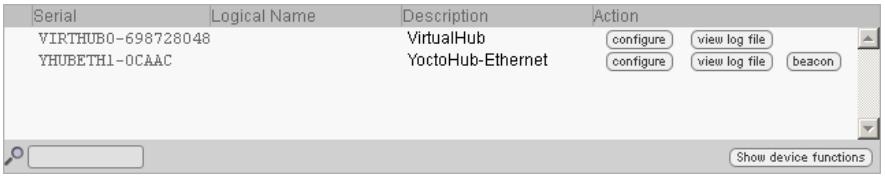
Si vous devez utiliser une configuration réseau particulière, vous pouvez configurer votre YoctoHub-Ethernet via son port de contrôle USB, en utilisant le *VirtualHub*[1].

Lancez un *VirtualHub* sur votre ordinateur favori et raccordez votre ordinateur au port *power / control port* du YoctoHub-Ethernet. Vous aurez besoin d'un câble USB A-MicroB. Raccordez aussi votre YoctoHub-Ethernet à un câble réseau opérationnel.




*Configuration: raccordez par USB votre YoctoHub-Ethernet à un ordinateur*

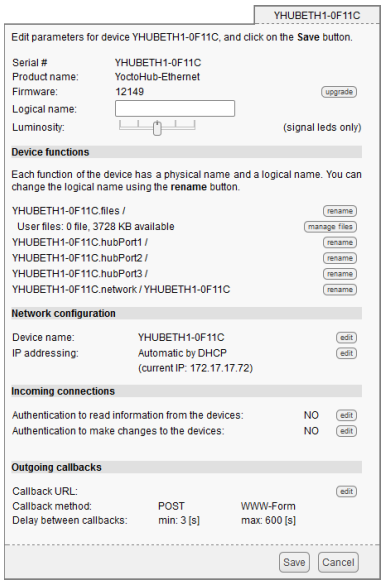
Lancez alors votre browser favori[2] sur l'url de votre *VirtualHub*. Il s'agit généralement *http://127.0.0.1:4444*. Vous obtiendrez la liste des modules Yoctopuce connectés par USB, dont votre YoctoHub-Ethernet




*Liste des modules Yoctopuce raccordés par USB à votre ordinateur, dont votre YoctoHub-Ethernet*

Cliquez sur le bouton **configure** correspondant à votre YoctoHub-Ethernet, votre obtiendrez la fenêtre de configuration du module. Cette fenêtre comporte une section **Network configuration**.

[1] *http://www.yoctopuce.com/FR/virtualhub.php*
[2] Sauf Opera

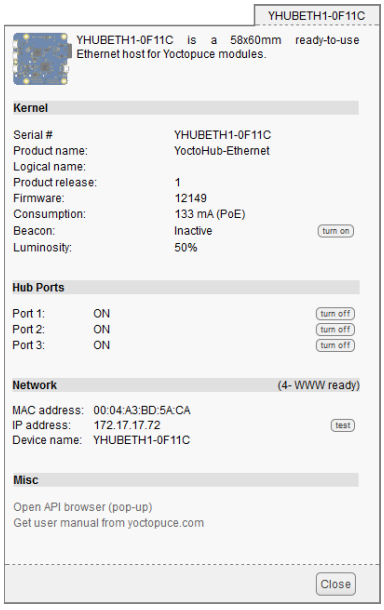



*Fenêtre de configuration du module YoctoHub-Ethernet*

Vous pouvez alors choisir si l'adresse IP de votre YoctoHub-Ethernet doit être attribuée par DHCP ou si elle doit être fixe. L'option DHCP est recommandée dans la mesure où cette fonctionnalité est supportée par la plupart des boîtiers ADSL (c'est la configuration par défaut). Si vous ne savez pas ce qu'est un serveur DCHP mais avez l'habitude de brancher des appareils sur votre réseau et de les voir marcher sans problème, ne touchez à rien.

Vous pouvez aussi choisir le nom réseau de votre YoctoHub-Ethernet. Vous pourrez ainsi accéder à votre YoctoHub-Ethernet en utilisant ce nom plutôt que son adresse IP. Une fois la partie réseau configurée, cliquez sur le bouton **Save**. Cela qui aura pour effet de sauver vos modifications et de fermer la fenêtre de configuration. Ces modifications étant sauvées dans la mémoire persistante du YoctoHub-Ethernet, il s'en rappellera même après avoir été privé de courant.

Cliquez sur le numéro de série correspondant à votre YoctoHub-Ethernet. Cela ouvrira la fenêtre détails de votre module:




*Les propriétés du YoctoHub-Ethernet*

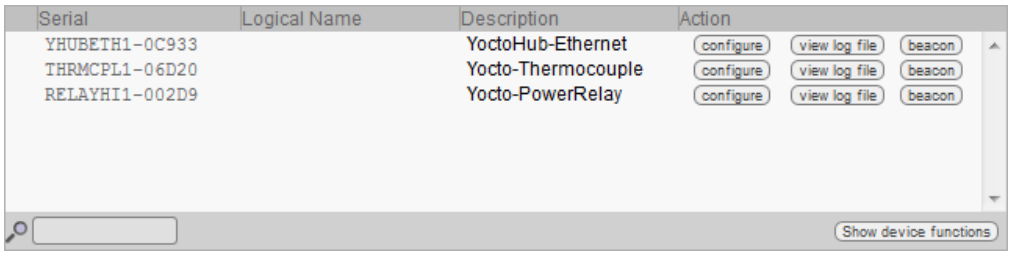
Cette fenêtre comporte une section qui relate l'état de la partie réseau du YoctoHub-Ethernet. Vous y trouverez son adresse MAC, adresse IP courante et nom de réseau. Cette section donne aussi l'état de la connexion réseau. Ces états peuvent être:

- 0- search for link: Le module cherche une liaison réseau. Si cet état persiste, il est probable que le câble réseau n'est pas branché
- 2- network linked: une liaison à été détectée.
- 3- LAN ready: le réseau local est opérationnel.
- 4- WWW ready: le module a pu vérifier la connectivité à Internet en se connectant à un serveur de temps (NTP).

Après avoir vérifié que votre module a bien une adresse IP valide, vous pouvez fermer la fenêtre détails, arrêter votre VirtualHub et débrancher le câble USB de contrôle de votre ordinateur: il suffira que le module soit alimenté pour l'utiliser.

Vous pouvez désormais accéder à votre YoctoHub-Ethernet en tapant directement son adresse IP dans la barre d'adresse de votre browser favori. Le module répond au port HTTP standard, mais aussi au port 4444 utilisé par le VirtualHub. Si l'adresse IP de votre module est *192.168.0.10*, vous pourrez donc le joindre avec l'URL *http://192.168.0.10*.

Vous obtiendrez alors directement l'interface du YoctoHub-Ethernet. Cette interface est en tout point identique à celle du *VirtualHub*. Vous retrouvez le YoctoHub-Ethernet sur la première ligne et les modules connectés au YoctoHub-Ethernet sur les suivantes.




*L'interface du YoctoHub-Ethernet est identique à celle d'un VirtualHub.*

Si vous avez attribué un nom à votre YoctoHub-Ethernet, vous pouvez aussi utiliser ce nom sur le réseau local. Par exemple, si vous avez utilisé le nom réseau *yoctohub*, vous pouvez contacter le module avec l'URL *http://yoctohub* sous Windows et avec l'URL *http://yoctohub.local* sous Mac OS X et Linux. Notez que cette technique est limitée au sous-réseau du YoctoHub-Ethernet. Si vous voulez contacter le module par nom depuis un autre réseau, vous devez utiliser une infrastructure DNS classique.

## 3.3. Configuration automatisée

Il est possible d'industrialiser la configuration réseau du YoctoHub-Ethernet. Vous trouverez dans les chapitres suivants de cette documentation la description des fonctions de programmation permettant de relire l'adresse Ethernet d'un module (adresse MAC), et de configurer tous ses paramètres réseau.

Les fonctions de configuration réseau sont aussi accessibles par ligne de commande, en utilisant l'utilitaire `YNetwork` disponible dans la librairie de programmation en ligne de commande[3].

Après avoir effectué un changement de réglage par programmation, prenez garde à appeler la fonction `saveToFlash()` pour vous assurez que les réglages soient sauvés de manière permanente dans la mémoire flash du module.

[3] http://www.yoctopuce.com/FR/libraries.php

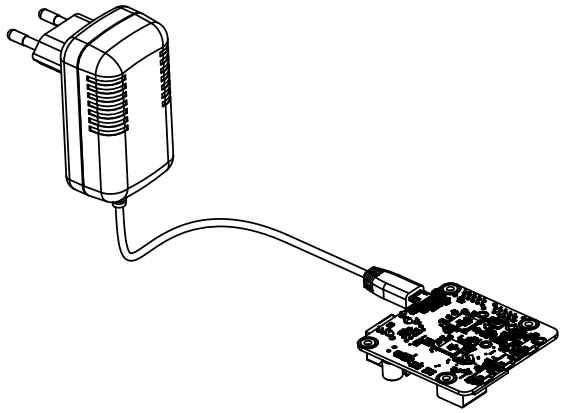
# 3.4. Connexions

## Alimentation

Le YoctoHub-Ethernet peut être alimenté de deux manières différentes:

### USB

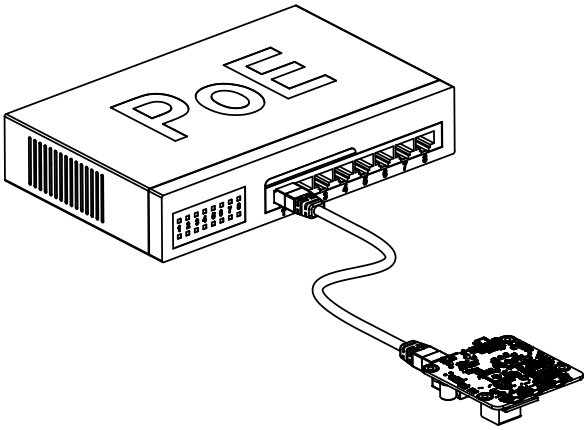
Branchez simplement un chargeur USB dans le port *power / control port*, assurez-vous tout de même que le chargeur soit d'une puissance électrique suffisante: le YoctoHub-Ethernet consomme environ 130mA, auxquels il faudra ajouter la consommation de chaque sous-module. Le YoctoHub-Ethernet est conçu pour gérer 2A au maximum, c'est pourquoi un chargeur USB capable de délivrer au moins 2A est recommandé. Par ailleurs, vous devrez veiller à ce que la consommation totale de l'ensemble hub + sous-modules ne dépasse pas cette limite.



*Le YoctoHub-Ethernet peut être alimenté par un chargeur USB*

### Power over Ethernet (PoE)

Votre YoctoHub-Ethernet est compatible *Power over Ethernet* (PoE). Cette technologie consiste à faire passer le courant d'alimentation directement dans le câble réseau. Pour cela vous avez besoin d'un équipement réseau qui alimente le câble Ethernet selon le standard 802.3af [4] Ces équipements se présentent généralement sous la forme d'un *Switch PoE*, combinant les fonctions "switch Ethernet" et "injecteur de puissance", ou d'un simple *Injecteur PoE*, permettant d'injecter la puissance sur un seul câble Ethernet. Il existe aussi des routeurs ADSL avec PoE, mais ceux-ci sont relativement rares. Si vous disposez d'un câble réseau relié à l'un de ces équipements, il vous suffit alors de connecter ce câble au YoctoHub-Ethernet pour que celui-ci soit immédiatement opérationnel.




*Le YoctoHub-Ethernet peut être alimenté par le câble Ethernet (PoE)*

Longtemps réservés au monde professionnel, ces équipements sont désormais facilement accessibles au grand public à des prix tout à fait raisonnables (quelques dizaines d'Euros pour un

[4] Le YoctoHub-Ethernet extrait la puissance du câble Ethernet à l'aide d'un composant *AG9705M*, qui implémente les deux alternatives A et B décrites dans le standard 802.3af. L'isolation galvanique entre le module et le réseau supporte 1500V.

injecteur, et un peu plus de 100 EUR pour un Switch PoE avec plusieurs ports alimentés). Les câbles réseau plats et fins sont généralement compatibles avec le standard PoE 802.3af.

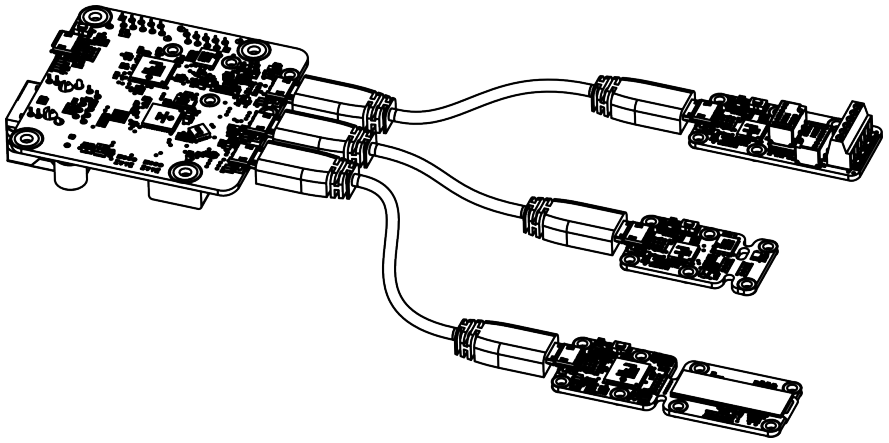
Le composant PoE du YoctoHub-Ethernet permet de délivrer environ 1.8 A à 5V. Si le YoctoHub-Ethernet détecte une surconsommation sur le PoE, il coupera automatiquement l'alimentation des ports descendants pour se protéger. Ces ports devront être réactivés explicitement, soit par logiciel soit grâce l'interface web du YoctoHub-Ethernet. La sur-consommation est physiquement signalée par la Led rouge *overload*.

Attention, la distribution de courant entre équipements PoE fait l'objet de négociations. Le YoctoHub-Ethernet va tenter de négocier 9W (1.8A) auprès de l'équipement qui fourni le courant sur le câble. Veillez à ne pas dépasser la puissance disponible sur l'équipement qui fournit le courant. Reportez-vous à la documentation de l'équipement en question pour plus de détails.

Il n'est pas recommandé de brancher en même temps un câble PoE et un chargeur USB sur votre YoctoHub-Ethernet. A priori, cela ne pose pas de problème majeur pour une utilisation très temporaire, mais en utilisation normale vous devez choisir l'une ou l'autre des sources d'alimentation.

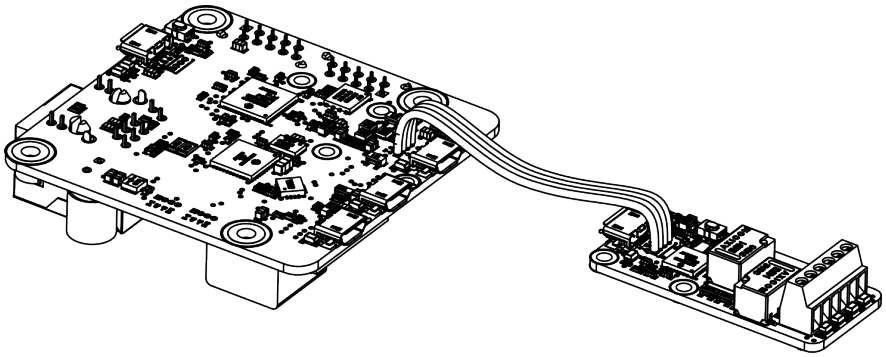
## Sous-modules

Le YoctoHub-Ethernet est capable de piloter tous les modules Yoctopuce de la gamme *Yocto*. Ces modules peuvent êtres connectés directement aux ports descendants, ils seront détectés automatiquement. Vous aurez besoin pour cela de câbles USB MicroB-MicroB. Vous pouvez utiliser des câbles OTG ou non, cela n'a pas d'importance.



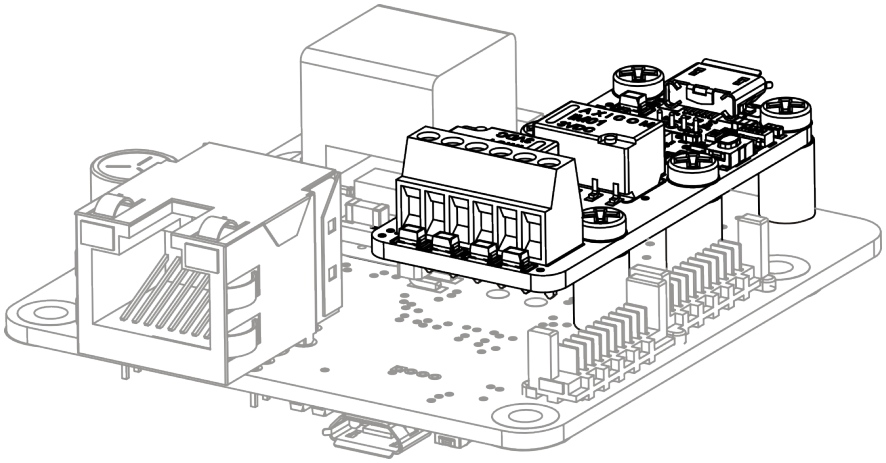
*Connexion des sous-modules à l'aide de câbles USB*

Alternativement, vous pouvez connecter vos modules de manière plus compacte à l'aide de câbles au pas 1.27mm: tous les modules Yoctopuce disposent en effet de contacts à cet effet. Vous pouvez soit souder des connecteurs 1.27mm sur les modules et utiliser des câbles avec connecteurs 1.27mm, soit souder directement du câble plat au pas 1.27mm. Si vous choisissez cette dernière option, il est recommandé d'utiliser du câble plat mono-brin, moins souple que le multi-brin mais beaucoup plus facile à souder. Soyez particulièrement attentif aux polarités: Le YoctoHub-Ethernet, tout comme l'ensemble de modules de la gamme Yoctopuce, n'est pas protégé contre les inversions de polarité. Une telle inversion a toutes les chances de détruire vos équipements. Assurez-vous que la position du contact carré de part et d'autre du câble correspondent.



*Connexion des sous-modules à l'aide de câble nappe*

Le YoctoHub-Ethernet est conçu pour que vous puissiez fixer un module simple largeur directement dessus. Vous aurez besoin de vis, d'entretoises[5] et d'un connecteur au pas 1.27mm[6]. Vous pouvez ainsi transformer un module Yoctopuce USB en en module réseau tout en gardant un format très compact.



*Fixation d'un module directement sur le hub*

Attention, le YoctoHub-Ethernet est conçu pour piloter des modules Yoctopuce uniquement. En effet le protocole utilisé entre le YoctoHub-Ethernet et les sous-modules n'est pas de l'USB mais un protocole propriétaire, beaucoup plus léger. Si d'aventure vous branchez un périphérique autre qu'un module Yoctopuce sur un des ports descendants du YoctoHub-Ethernet, le port en question sera automatiquement désactivé pour éviter d'endommager le périphérique.

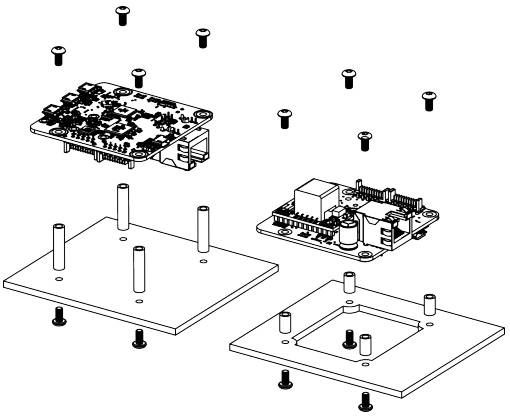
[5] http://www.yoctopuce.com/FR/products/accessoires-et-connectique/fix-2-5mm
[6] http://www.yoctopuce.com/FR/products/accessoires-et-connectique/board2board-127

# 4. Montage

Ce chapitre fournit des explications importantes pour utiliser votre module YoctoHub-Ethernet en situation réelle. Prenez soin de le lire avant d'aller trop loin dans votre projet si vous voulez éviter les mauvaises surprises.

## 4.1. Fixation

Pendant la mise au point de votre projet, vous pouvez vous contenter de laisser le hub se promener au bout de son câble. Veillez simplement à ce qu'il ne soit pas en contact avec quoi que soit de conducteur (comme vos outils). Une fois votre projet pratiquement terminé, il faudra penser à faire en sorte que vos modules ne puissent pas se promener à l'intérieur.



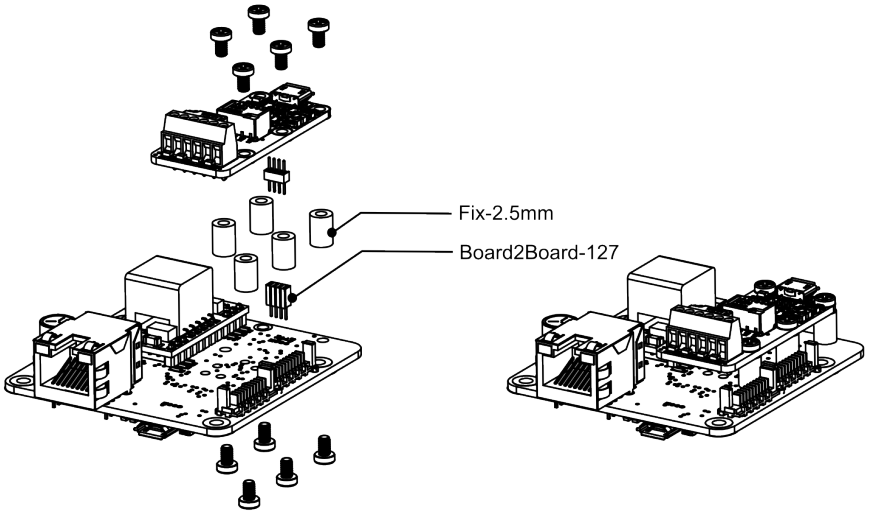
*Exemples de montage sur un support.*

Le module YoctoHub-Ethernet dispose de trous de montage 3mm. Vous pouvez utiliser ces trous pour y passer des vis. Le diamètre de la tête de ces vis ne devra pas dépasser 8mm, sous peine d'endommager les circuits du module.

Veillez à que l'électronique module ne soit pas en contact avec le support. La méthode recommandée consiste à utiliser des entretoises. Vous pouvez monter le module dans le sens qui vous convient: mais vous devez conscient du fait que les composants électroniques du YoctoHub-Ethernet, la partie réseau en particulier, dégagent de la chaleur. Vous devrez donc faire en sorte que la chaleur ne puisse pas s'accumuler.

## 4.2. Fixation d'un sous-module

Le YoctoHub-Ethernet est conçu pour que vous puissiez visser un module simple largeur directement dessus. Par simple largeur, on entend les modules de 20 mm de large. Tous les modules simple largeur ont leurs 5 trous de fixation et le connecteur USB au même endroit. Le sous-module peut être fixé à l'aide de vis et d'entretoises. Il y a derrière les connecteurs USB du YoctoHub-Ethernet et du sous-module un ensemble de 4 contacts qui permettent d'effectuer la connexion électrique entre le hub et le sous-module. Si vous ne vous sentez pas suffisamment à l'aise avec un fer à souder, vous pouvez aussi aussi utiliser un simple câble USB MicroB-MicroB, OTG ou non.



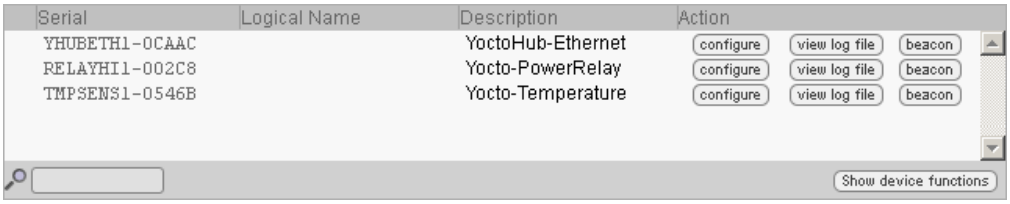

*Fixation d'un module directement sur le hub*

Prenez garde à bien monter le module sur la face prévue, comme illustré ci-dessus. Les 5 trous du module doivent correspondre aux 5 trous du YoctoHub-Ethernet, et le contact carré sur sur le module doit être connecté au contact carré sur le port descendant du YoctoHub-Ethernet. Si vous montez un module sur l'autre face ou d'une autre manière, la polarité du connecteur sera inversée et vous risquez fort d'endommager définitivement votre matériel.

Tous les accessoires nécessaires à la fixation d'un module sur votre YoctoHub-Ethernet sont relativement courants. Vous pourrez les trouver sur le site de Yoctopuce tout comme sur la plupart des sites vendant du matériel électronique. Attention cependant, la tête des vis servant à fixer le sous-module devra avoir un diamètre maximum de 4.5 millimètres, sous peine d'endommager les composants électroniques.

# 5. Utilisation du YoctoHub-Ethernet

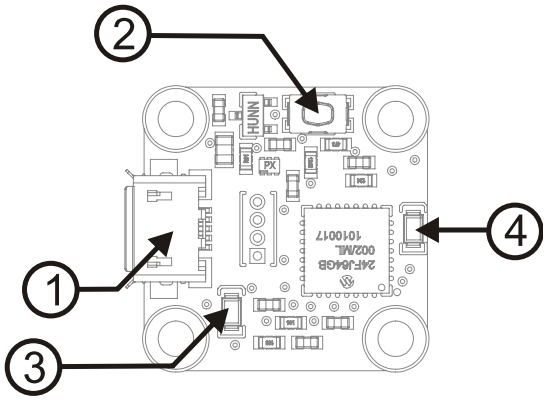
Outre fournir un accès réseau au module Yoctopuce, le YoctoHub-Ethernet permet de tester et configurer vos modules Yoctopuce. Pour ce faire connectez-vous sur votre YoctoHub-Ethernet à l'aide de votre navigateur internet favori[1]. Utilisez l'adresse IP du YoctoHub-Ethernet ou encore son nom réseau, par exemple `http://192.168.0.10`. Une liste comprenant votre YoctoHub-Ethernet ainsi que les modules qui lui sont connectés devrait apparaître.




*Interface Web du YoctoHub-Ethernet.*

## 5.1. Localisation des modules

L'interface principale vous montre une ligne par module connecté, si vous avez plusieurs modules du même modèle, vous pouvez localiser un module particulier en cliquant sur le bouton **beacon** correspondant: cela aura pour effet de faire clignoter la Led bleue du module et d'afficher sur l'interface une pastille bleue au début de la ligne correspondante. Vous pouvez faire la même manipulation en appuyant sur le Yocto-bouton d'un module connecté.

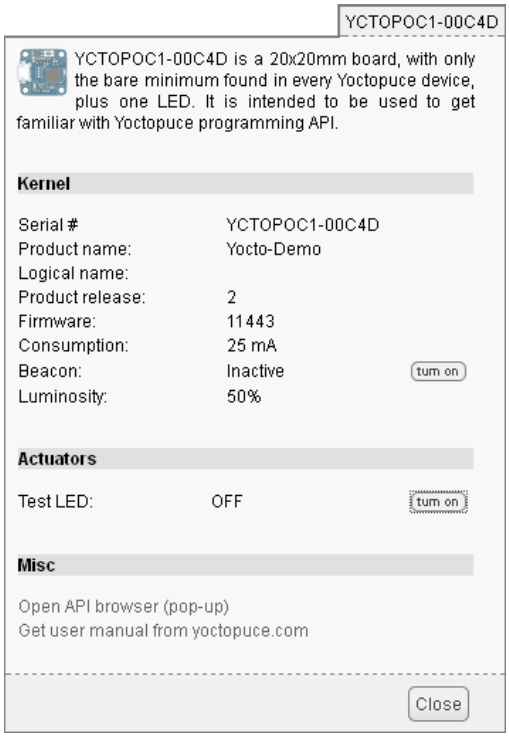



*Yocto-bouton(1) et Led(2) de localisation d'un module Yocto-Demo. Ces deux éléments sont généralement placés au même endroit sur les modules.*

[1] L'interface du YoctoHub-Ethernet est régulièrement testée sur Internet Explorer, Firefox, Chrome et Safari. Elle ne fonctionne pas avec Opéra

## 5.2. Test des modules

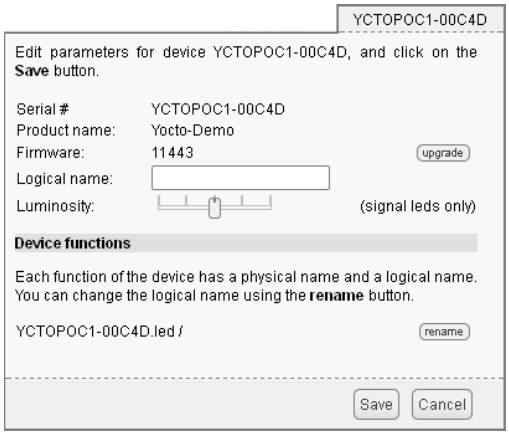
Pour tester un module, cliquez simplement sur le numéro de série d'un module dans l'interface, une fenêtre spécifique au module s'ouvrira. Cette fenêtre permet généralement d'activer les fonctions principales du module. Reportez-vous au manuel du module correspondant pour plus de détails.[2]




*Fenêtre "détails" du module Yocto-Demo, obtenue via l'interface du YoctoHub-Ethernet.*

## 5.3. Configuration des modules

Vous pouvez configurer un module en cliquant sur le bouton **Configure** correspondant dans l'interface principale, une fenêtre spécifique au module s'ouvre alors. Cette fenêtre permet au minimum de donner un nom logique au module ainsi que de mettre à jour son firmware. Reportez-vous au manuel du module correspondant pour plus de détails.



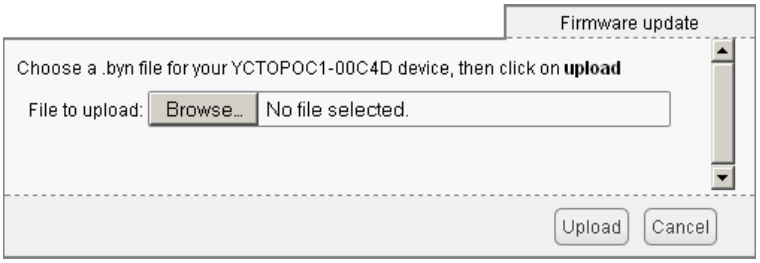

*Fenêtre "configure" du module Yocto-Demo.*

[2] Vous n'êtes pas obligé d'avoir un YoctoHub-Ethernet plus récent qu'un module pour le tester/configurer: tous les éléments spécifiques aux interfaces des modules sont stockés dans la ROM des modules, et non pas dans le YoctoHub-Ethernet.

## 5.4. Upgrades des firmwares

Les modules Yoctopuce sont en fait de véritables ordinateurs, ils contiennent même un petit serveur Web. Et comme tous les ordinateurs, il est possible de mettre à jour leur logiciel de contrôle (firmware). Des nouveaux firmwares pour chaque module sont régulièrement publiés, ils permettent généralement d'ajouter de nouvelles fonctionnalités au module, et/ou de corriger d'éventuels bugs[3].

Pour mettre à jour le firmware d'un module, vous devez d'abord vous procurer le firmware, il peut être téléchargé depuis la page produit du module sur le site de Yoctopuce [4]. L'interface propose aussi un lien direct si elle détecte que le firmware n'est pas à jour[5]. Ces firmwares se présentent sous la forme de fichiers `.byn` de quelques dizaines de Kilo-octets, sauvez celui qui vous intéresse sur votre disque local.




*Fenêtre de mise à jour du firmware.*

Une fois votre fichier de firmware disponible localement, ouvrez la fenêtre **configuration** d'un module et cliquez sur le bouton **upgrade**. L'interface va vous demander de choisir le fichier de firmware que vous désirez utiliser. Entrez le nom du fichier et cliquez sur **Upload**. A partir de là, tout est automatique, le YoctoHub-Ethernet va faire redémarrer le module en mode "mise à jour", mettre à jour le firmware, puis redémarrer le module en mode normal. Les réglages de configuration du module seront préservés. Ne débranchez pas le module pendant la procédure de mise à jour.

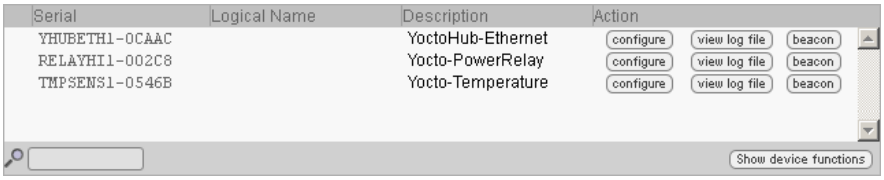
Le firmware du YoctoHub-Ethernet peut être mis à jour de la même manière

En cas de perte de contrôle pendant une mise à jour du firmware (panne de courant ou débranchement involontaire), il est toujours possible de forcer manuellement un rechargement du firmware, même si le sous-module n'apparaît même plus dans la fenêtre du YoctoHub-Ethernet. Pour ce faire, débranchez le sous-module et rebranchez-le en maintenant le Yocto-bouton pressé. Le module démarrera alors dans le mode de mise à jour du firmware, et vous pourrez recommencer la procédure de chargement.

[3] Ne faites jamais confiance à des gens qui vous disent que leur logiciel n'a pas de bug :-)
[4] www.yoctopuce.com
[5] A condition qu'elle ait réussi à accéder au site Web de Yoctopuce

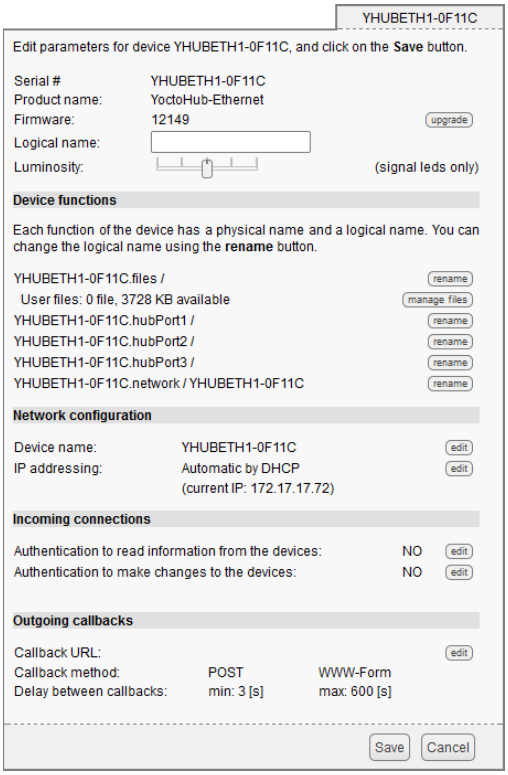
# 6. Contrôle d'accès

Le YoctoHub-Ethernet vous permet d'instaurer un contrôle d'accès à vos modules Yoctopuce. Pour ce faire, cliquez simplement sur le bouton **Configure** de la ligne correspondant au YoctoHub-Ethernet dans l'interface.




*Cliquez sur le bouton "Configure" de la première ligne*

Cela aura pour effet de faire apparaître la fenêtre de configuration du YoctoHub-Ethernet.




*La fenêtre de configuration du YoctoHub-Ethernet*

Ce contrôle d'accès est contrôlé depuis la section **Incoming connections**. Il peut se faire à deux niveaux distincts.

## 6.1. Accès "admin" protégé

Le mot de passe *admin* verrouille les accès en écriture sur les modules. Lorsqu'il est configuré, seuls les accès de type *admin* permettent d'accéder aux modules en lecture et en écriture. Les utilisateurs utilisant le login *admin* pourront éditer la configuration des modules vus par ce YoctoHub-Ethernet comme ils le souhaitent.

## 6.2. Accès "user" protégé

Le mot de passe *user* verrouille toute utilisation des modules. Lorsqu'il est configuré, toute utilisation sans mot de passe devient impossible. Les accès de type *user* ne permettent d'accéder aux modules qu'en lecture seule c'est-à-dire seulement pour consulter l'état des modules. Si vous instaurez simultanément un contrôle d'accès de type *user* et de type *admin*, les utilisateurs utilisant le login *user* ne pourront pas modifier la configuration des modules vus par ce YoctoHub-Ethernet.

Si vous configurez un accès *admin*, sans configurer d'accès *user*, les utilisateurs pourront continuer à consulter vos modules en lecture sans avoir à entrer de mot de passe.

Pour configurer l'accès au YoctoHub-Ethernet, cliquez sur le bouton **edit** des lignes **Authentication to read the information from the devices** ou **Authentication to write information to the devices**

## 6.3. Influence sur les API

Attention, le contrôle d'accès agira aussi sur les API Yoctopuce qui tenteront de se connecter à ce YoctoHub-Ethernet. Dans les API Yoctopuce, la gestion des droits d'accès est réalisée au niveau de l'appel à la fonction `RegisterHub()`: vous devrez donner l'adresse du YoctoHub-Ethernet sous la forme *login:password@adresse:port*, par exemple:

```
yRegisterHub("admin:mypass@192.168.0.10:4444",errmsg);
```

## 6.4. Effacement des mots de passe

Si vous perdez le mot passe de votre YoctoHub-Ethernet, vous pouvez reprendre le contrôle de votre module en réinitialisant tous ses réglages à la valeur par défaut. Pour ce faire, procurez-vous un câble USB pour le YoctoHub-Ethernet, et branchez-le à un ordinateur avec le VirtualHub[1] installé en maintenant le Yocto-bouton pressé. Ceci va forcer le YoctoHub-Ethernet à démarrer en mode de mise à jour du firmware. Il apparaît alors dans le VirtualHub en dessous de la liste des modules. Cliquez sur son numéro de série et choisissez un fichier de firmware à charger sur le module. Une fois le firmware rechargé avec cette méthode, le module sera réinitialisé avec les réglages d'usine, sans contrôle d'accès.

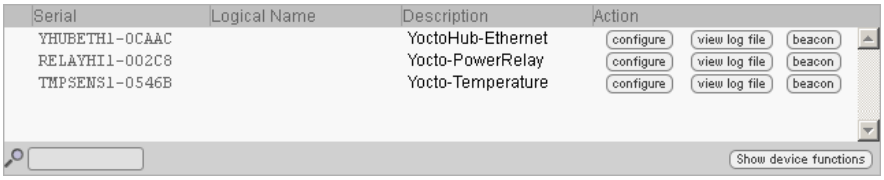
[1] http://www.yoctopuce.com/FR/virtualhub.php

# 7. Interactions avec l'extérieur

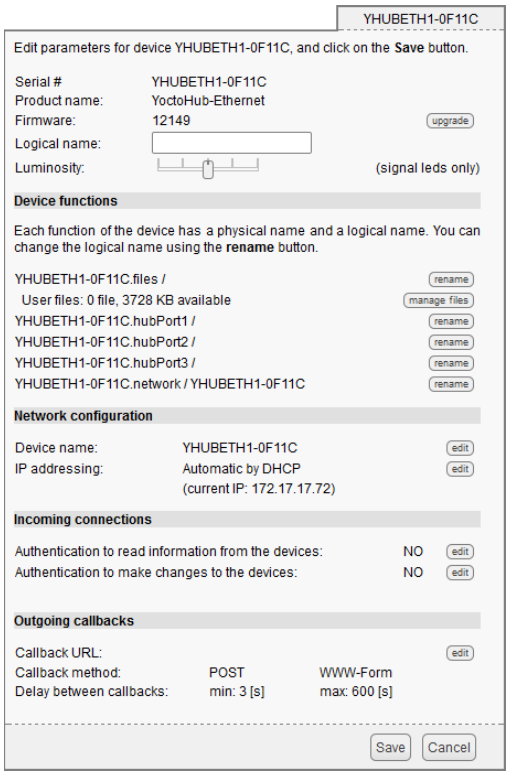
Le YoctoHub-Ethernet est capable de poster sur le site web de votre choix l'état des modules qu'il voit. Les valeurs sont postées à intervalles réguliers et à chaque fois qu'une valeur change de manière significative. Cette fonctionnalité vous permettra d'interfacer vos modules Yoctopuce avec divers services web.

## 7.1. Configuration

Pour utiliser cette fonctionnalité, cliquez simplement sur le bouton **Configure** de la ligne correspondant au YoctoHub-Ethernet dans l'interface, puis cliquez sur le bouton **edit** de la section **Outgoing callback**.




*Cliquez sur le bouton "Configure" de la première ligne*

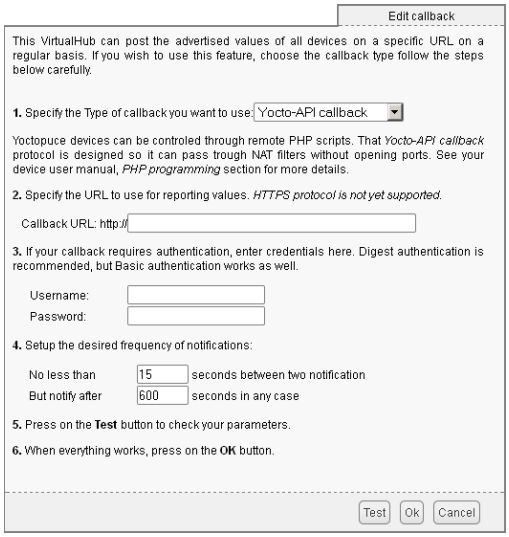



*Puis éditez la section Outgoing callbacks.*

La fenêtre de configuration des callbacks apparaît. Cette fenêtre va vous permettre de définir comment votre YoctoHub-Ethernet va pouvoir interagir avec un serveur Web externe. Vous avez plusieurs type d'interactions a votre disposition.

## 7.2. User defined callback

Les "*User defined callback*" vous permettent de personnaliser la manière dont votre YoctoHub-Ethernet va interagir avec un site Web externe. Vous avez besoin de définir l'URL du serveur Web sur lequel le YoctoHub-Ethernet va poster l'état de ses devices. Notez que seul le protocole HTTP est supporté (pas de HTTPS).


Edit callback
This VirtualHub can post the advertised values of all devices on a specific URL on a regular basis. If you wish to use this feature, choose the callback type follow the steps below carefully.
1. Specify the Type of callback you want to use:
Yocto-API callback
Yoctopuce devices can be controled through remote PHP scripts. That Yocto-API callback protocol is designed so it can pass trough NAT filters without opening ports. See your device user manual, PHP programming section for more details.
2. Specify the URL to use for reporting values. HTTPS protocol is not yet supported.
Callback URL: http://
3. If your callback requires authentication, enter credentials here. Digest authentication is recommended, but Basic authentication works as well.
Username:
Password:
4. Setup the desired frequency of notifications:
No less than
15
seconds between two notification
But notify after
600
seconds in any case
5. Press on the Test button to check your parameters.
6. When everything works, press on the OK button.
Test
Ok
Cancel

*La fenêtre de configurations des callbacks*

Si vous désirez protéger votre script de callback, vous pouvez configurer un contrôle d'accès HTTP standard sur le serveur Web. Le YoctoHub-Ethernet sait comment gérer les méthodes standard d'identification de HTTP: indiquez simplement le nom d'utilisateur et le mot de passe nécessaires pour accéder à la page. Il est possible d'utiliser la méthode "Basic" aussi bien que la méthode

"Digest", mais il est recommandé d'utiliser la méthode "Digest", car elle est basée sur un protocole de question-réponse qui évite la transmission du mot de passe sur le réseau et évite aussi les copies d'autorisation.

Le YoctoHub-Ethernet poste avec la méthode POST les valeurs notifiées[1] des modules à intervalle régulier, et à chaque fois qu'une de ces valeurs change de manière significative. Vous pouvez changer les délais d'attente entre les posts.

## Tests

Afin de vous permettre de déboguer le processus, le YoctoHub-Ethernet vous permet de visualiser la réponse au callback envoyé par le serveur Web. Cliquez simplement sur le bouton **test** une fois que vous avez renseigné tous les champs. Si le résultat vous paraît satisfaisant, fermez la fenêtre de debug, et cliquez sur **Ok**.

## Formats

Les valeurs sont postées sous une des formes suivantes:

1. Si un nom logique a été défini pour une fonction:

```
NOM_LOGIQUE_DE_LA_FONCTION = VALEUR
```

2. Si un nom logique a été défini pour le module, mais pas pour la fonction:

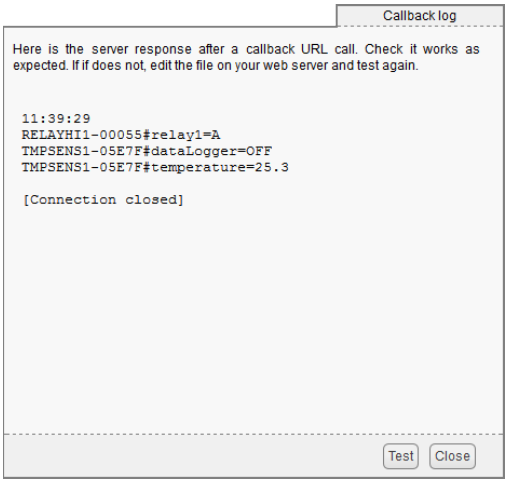
```
NOM_DU_MODULE#NOM_HARDWARE = VALUE
```

3. Si aucun nom logique n'a été attribué:

```
NUMERO_DE_SERIE#NOM_HARDWARE = VALEUR
```

Voici un script PHP qui vous permettra de visualiser le contenu des données postées par le callback, suivi du résultat dans la fenêtre de debug.

```php
<?php
  Print(Date('H:i:s')."\r\n");
  foreach ($_POST as $key=>$value) {
      Print("$key=$value\r\n");
  }
?>
```


Callback log
Here is the server response after a callback URL call. Check it works as expected. If if does not, edit the file on your web server and test again.
11:39:29
RELAYHI1-00055#relay1=A
TMPSENS1-05E7F#dataLogger=OFF
TMPSENS1-05E7F#temperature=25.3
[Connection closed]
Test
Close

*Le résultat du test de callback avec un Yocto-PowerRelay et un Yocto-Temperature.*

[1] Les valeurs notifiées sont celles que vous voyez quand vous cliquez sur *show functions* dans l'interface principale du YoctoHub-Ethernet .

## 7.3. Yocto-API callback

L'API Yoctopuce PHP est capable de fonctionner en mode *callback*. Dans ce mode un script PHP peut prendre le contrôle de vos modules à travers un filtre NAT sans que vous ayez à ouvrir un port. Typiquement cela permet de contrôler depuis un site Web public des modules Yoctopuce installés derrière un router ADSL privé. Le YoctoHub-Ethernet sert alors de relais. Vous avez simplement à définir l'URL du script PHP de contrôle et éventuellement les crédentiels nécessaires pour y accéder. Vous trouvez plus d'informations sur le fonctionnement du mode "callback" de l'API PHP dans le mode d'emploi de vos modules Yoctopuce.

## 7.4. Xively (anciennement Cosm)

Xively[2] est un service de cloud payant (mais avec un mode de développement limité gratuit) qui vous permettra tracer des graphes avec les données issues de vos capteurs Yoctopuce, et ce sans écrire la moindre ligne de code. Vous avez besoin d'un compte Xively et de définir un feed et une clef API sur le site de Xively. Entrez ces deux paramètres dans l'interface du YoctoHub-Ethernet , c'est tout. Au besoin vous trouverez des explications plus détaillées sur le blog de Yoctopuce[3]. Yoctopuce n'est en aucune manière affilié à Xively.

## 7.5. Thinkspeak

ThingSpeak[4] est un autre service de cloud, entièrement gratuit lui, qui permet aussi de tracer des graphes avec des données issues de vos capteurs Yoctopuce. Ce service présente quelques limitations de fonctionnalité par rapport à Xively, mais a l'avantage de ne nécessiter aucune license payante. Vous trouverez sur le blog de Yoctopuce[5] comment configurer votre YoctoHub-Ethernet pour poster des données directement sur ThingSpeak. Yoctopuce n'est en aucune manière affilié à ThingSpeak.

[2] www.xively.com
[3] http://www.yoctopuce.com/FR/article/enregistrer-des-mesures-sur-internet
[4] www.thingspeak.com
[5] http://www.yoctopuce.com/FR/article/alternatives-a-cosm-pour-enregistrer-des-mesures

# 8. Programmation

## 8.1. Accès aux modules connectés

Le YoctoHub-Ethernet se comporte exactement comme un ordinateur faisant tourner un *VirtualHub*. La seule différence entre un programme utilisant l'API Yoctopuce utilisant des modules en USB natif et ce même programme utilisant des modules Yoctopuce connecté à un YoctoHub-Ethernet se situe au niveau de l'appel a *registerHub*. Pour utiliser des modules USB connectés en natif, le paramètre de *RegisterHub* est *usb*, pour utiliser des modules connectés à un YoctoHub-Ethernet, il suffit de remplacer ce paramètre par l'adresse IP du YoctoHub-Ethernet. Par en exemple, en Delphi:

```
YRegisterHub("usb",errmsg);
```

devient

```
YRegisterHub("192.168.0.10",errmsg); // l'adresse IP du hub est 192.168.0.10
```

## 8.2. Contrôle du YoctoHub-Ethernet

Du point de vue API de programmation, le YoctoHub-Ethernet est un module comme les autres. Il est parfaitement contrôlable depuis l'API Yoctopuce. Pour ce faire, vous aurez besoin des classes suivantes.

### Module

Cette classe, commune à tous les modules Yoctopuce permet de contrôler le module en temps que tel. Elle vous permettra de contrôler la Yocto-Led, de connaître la consommation sur USB du YoctoHub-Ethernet, etc.

### Network

Cette classe permet de contrôler la partie réseau du YoctoHub-Ethernet, vous pourrez contrôler l'état du link, lire l'adresse MAC, changer l'adresse IP du YoctoHub-Ethernet, connaître la consommation sur PoE, etc.

### HubPort

Cette classe permet de contrôler la partie hub. Vous pourrez activer ou désactiver les ports du YoctoHub-Ethernet, vous pourrez aussi savoir quel module est connecté à quel port.

## Files

Cette classe permet d'accéder aux fichiers stockées dans la mémoire flash du YoctoHub-Ethernet. Le YoctoHub-Ethernet dispose en effet d'un petit système de fichiers qui vous permet de stocker par exemple une Web App contrôlant les modules connectés au YoctoHub-Ethernet.

Vous trouverez quelques exemples de contrôle du YoctoHub-Ethernet par programmation dans les librairies Yoctopuce, disponibles gratuitement sur le site de Yoctopuce.

# 9. Personnalisation de l'interface Web

Votre YoctoHub-Ethernet dispose d'un petit système de fichiers embarqué, qui permet de stocker des fichiers personnalisés utilisables par le YoctoHub-Ethernet. Le système de fichiers se manipule grâce à la libraire *yocto_files*. Vous pourrez y stocker les fichiers de votre choix. Au besoin, vous pourrez y stocker une application Web permettant de gérer les modules connectés à votre YoctoHub-Ethernet.

## 9.1. Utilisation

### Utilisation interactive

L'interface Web du YoctoHub-Ethernet fournit une interface sommaire pour manipuler le contenu du système de fichiers: cliquez simplement sur le bouton *configuration* correspondant à votre module dans l'interface du hub, puis sur le bouton *manage files*. Les fichiers présents sont listés, et vous pouvez les visualiser, les effacer ou en ajouter (téléchargement).

En raison de sa petite taille, le système de fichiers ne possède pas de notion explicite de répertoire. Vous pouvez toutefois utiliser la barre oblique "/" à l'intérieur des noms de fichiers pour les *classer* comme s'ils étaient dans des répertoires.

### Utilisation programmée

Le système de fichiers s'utilise avec la librairie *yocto_files*. Les fonctions de bases sont disponibles:

- *upload* vous permet de créer un nouveau fichier sur le module, dont vous fournissez le contenu;
- *get_list* vous permet de connaître la liste de fichier présents sur le module, y compris la taille et le CRC32 du contenu;
- *download* vous permet de le récupérer dans une variable le contenu d'un fichier présent sur le module;
- *remove* permet d'effacer un fichier du module.
- *format* permet de réinitialiser le système de fichiers à un état vide, non fragmenté.

Un programme utilisant le système de fichier bien conçu devrait toujours commencer par s'assurer que les fichiers nécessaires à son fonctionnement sont présents sur le module, et si nécessaire les charger sur le module. Cela permet de gérer de manière transparente les mises à jour logicielles et le déploiement de l'application sur des nouveaux modules. Pour faciliter la détection des versions de fichiers présents sur le module, la méthode *get_list* retourne pour chaque fichier une signature sur 32 bit appelée CRC (Cyclic Redundancy Check), qui identifie de manière fiable le contenu du fichier. Ainsi, si le CRC du fichier correspond, il y a moins d'une chance sur 4 milliards que son contenu ne soit pas le bon. Vous pouvez même calculer dans votre programme par avance le CRC du contenu

que vous désirez, et ainsi le vérifier sans avoir à transférer le fichier. La fonction CRC utilisée par le système de fichiers Yoctopuce est la même que celle d'Ethernet, Gzip, PNG, etc. Sa valeur caractéristique pour la chaîne de neuf caractères "123456789" est 0xCBF43926.

### Utilisation par HTTP

Les fichiers que vous avez chargés sur sur votre YoctoHub-Ethernet sont accessibles par HTTP, à la racine du module (au même niveau que l'API REST). Cela permet de charger par exemple des pages d'interface HTML et Javascript personnalisées. Vous ne pouvez toutefois pas remplacer le contenu d'un fichier préchargé sur le module, mais seulement en ajouter des nouveaux.

## 9.2. Limitations

Le filesystem embarqué sur votre YoctoHub-Ethernet a quelques limitations techniques:

- Son espace de stockage maximal est 3.5Mo, répartis en blocs permettant de stocker jusqu'à environ 800 fichiers
- L'effacement d'un fichier ne récupère pas nécessairement immédiatement la totalité de la place utilisée par le fichier. L'espace non libéré sera entièrement réutilisé si l'on recrée un fichier du même nom, mais pas forcément si l'on crée des fichiers utilisant chaque fois des noms différents. Pour cette raison, il n'est pas recommandé de générer automatiquement des fichiers avec des noms toujours différents.
- L'espace non libéré peut être entièrement récupéré avec la commande *format*, qui libère la totalité des fichiers.
- Chaque mise à jour du firmware provoque implicitement un formattage complet du filesystem.
- Comme toutes les mémoires flash, la mémoire utilisée pour stocker les fichiers a une durée de vie de 100'000 cycles d'effacement environ. C'est assez, mais ce n'est pas illimité. Prenez donc garde à ne pas écrire et effacer inutilement des fichiers en boucle très rapidement, sous peine de détruire votre module.

# 10. Référence de l'API de haut niveau

Ce chapitre résume les fonctions de l'API de haut niveau pour commander votre YoctoHub-Ethernet. La syntaxe et les types précis peuvent varier d'un langage à l'autre mais, sauf avis contraire toutes sont disponibles dans chaque language. Pour une information plus précise sur les types des arguments et des valeurs de retour dans un langage donné, veuillez vous référer au fichier de définition pour ce langage (`yocto_api.*` ainsi que les autres fichiers `yocto_*` définissant les interfaces des fonctions).

Dans les langages qui supportent les exceptions, toutes ces fonctions vont par défaut générer des exceptions en cas d'erreur plutôt que de retourner la valeur d'erreur documentée pour chaque fonction, afin de faciliter le déboguage. Il est toutefois possible de désactiver l'utilisation d'exceptions à l'aide de la fonction `yDisableExceptions()`, si l'on préfère travailler avec des valeurs de retour d'erreur.

Ce chapitre ne reprend pas en détail les concepts de programmation des modules Yoctopuce. Vous trouverez des explications plus détaillées dans la documentation des modules que vous souhaitez raccorder à votre YoctoHub-Ethernet.

# 10.1. Interface d'un port de Yocto-hub

Les objets YHubPort permettent de contrôler l'alimentation des ports d'un YoctoHub, ainsi que de détecter si un module y est raccordé et lequel. Un YHubPort reçoit toujours automatiquement comme nom logique le numéro de série unique du module Yoctopuce qui y est connecté.

Pour utiliser les fonctions décrites ici, vous devez inclure:

```
js      <script type='text/javascript' src='yocto_hubport.js'></script>
nodejs  var yoctolib = require('yoctolib');
        var YHubPort = yoctolib.YHubPort;
php     require_once('yocto_hubport.php');
cpp     #include "yocto_hubport.h"
m       #import "yocto_hubport.h"
pas     uses yocto_hubport;
vb      yocto_hubport.vb
cs      yocto_hubport.cs
java    import com.yoctopuce.YoctoAPI.YHubPort;
py      from yocto_hubport import *
```

| Fonction globales |
|---|
| **yFindHubPort**(**func**)<br>Permet de retrouver un port de Yocto-hub d'après un identifiant donné. |
| **yFirstHubPort**()<br>Commence l'énumération des port de Yocto-hub accessibles par la librairie. |
| **Méthodes des objets YHubPort** |
| **hubport→describe**()<br>Retourne un court texte décrivant le port de Yocto-hub au format TYPE(NAME)=SERIAL.FUNCTIONID. |
| **hubport→get_advertisedValue**()<br>Retourne la valeur courante du port de Yocto-hub (pas plus de 6 caractères). |
| **hubport→get_baudRate**()<br>Retourne la vitesse de transfert utilisée par le port de Yocto-hub, en kbps. |
| **hubport→get_enabled**()<br>Retourne vrai si le port du Yocto-hub est alimenté, faux sinon. |
| **hubport→get_errorMessage**()<br>Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du port de Yocto-hub. |
| **hubport→get_errorType**()<br>Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du port de Yocto-hub. |
| **hubport→get_friendlyName**()<br>Retourne un identifiant global du port de Yocto-hub au format NOM_MODULE.NOM_FONCTION. |
| **hubport→get_functionDescriptor**()<br>Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction. |
| **hubport→get_functionId**()<br>Retourne l'identifiant matériel du port de Yocto-hub, sans référence au module. |
| **hubport→get_hardwareId**()<br>Retourne l'identifiant matériel unique du port de Yocto-hub au format SERIAL.FUNCTIONID. |
| **hubport→get_logicalName**() |

Retourne le nom logique du port de Yocto-hub.

**hubport→get_module**()

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**hubport→get_module_async**(**callback**, **context**)

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**hubport→get_portState**()

Retourne l'état actuel du port de Yocto-hub.

**hubport→get_userData**()

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

**hubport→isOnline**()

Vérifie si le module hébergeant le port de Yocto-hub est joignable, sans déclencher d'erreur.

**hubport→isOnline_async**(**callback**, **context**)

Vérifie si le module hébergeant le port de Yocto-hub est joignable, sans déclencher d'erreur.

**hubport→load**(**msValidity**)

Met en cache les valeurs courantes du port de Yocto-hub, avec une durée de validité spécifiée.

**hubport→load_async**(**msValidity**, **callback**, **context**)

Met en cache les valeurs courantes du port de Yocto-hub, avec une durée de validité spécifiée.

**hubport→nextHubPort**()

Continue l'énumération des port de Yocto-hub commencée à l'aide de `yFirstHubPort()`.

**hubport→registerValueCallback**(**callback**)

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

**hubport→set_enabled**(**newval**)

Modifie le mode d'activation du port du Yocto-hub.

**hubport→set_logicalName**(**newval**)

Modifie le nom logique du port de Yocto-hub.

**hubport→set_userData**(**data**)

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

**hubport→wait_async**(**callback**, **context**)

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YHubPort.FindHubPort() YHubPort yFindHubPort()

Permet de retrouver un port de Yocto-hub d'après un identifiant donné.

```
js      function yFindHubPort( func)
nodejs  function FindHubPort( func)
php     function yFindHubPort( $func)
cpp     YHubPort* yFindHubPort( const string& func)
m       YHubPort* yFindHubPort( NSString* func)
pas     function yFindHubPort( func: string): TYHubPort
vb      function yFindHubPort( ByVal func As String) As YHubPort
cs      YHubPort FindHubPort( string func)
java    YHubPort FindHubPort( String func)
py      def FindHubPort( func)
```

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le port de Yocto-hub soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YHubPort.isOnline()` pour tester si le port de Yocto-hub est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

**Paramètres :**

**func** une chaîne de caractères qui référence le port de Yocto-hub sans ambiguïté

**Retourne :**

un objet de classe `YHubPort` qui permet ensuite de contrôler le port de Yocto-hub.

## YHubPort.FirstHubPort() YHubPort yFirstHubPort()

Commence l'énumération des port de Yocto-hub accessibles par la librairie.

```
js      function yFirstHubPort( )
nodejs  function FirstHubPort( )
php     function yFirstHubPort( )
cpp     YHubPort* yFirstHubPort( )
m       YHubPort* yFirstHubPort( )
pas     function yFirstHubPort( ): TYHubPort
vb      function yFirstHubPort( ) As YHubPort
cs      YHubPort FirstHubPort( )
java    YHubPort FirstHubPort( )
py      def FirstHubPort( )
```

Utiliser la fonction `YHubPort.nextHubPort()` pour itérer sur les autres port de Yocto-hub.

**Retourne :**

un pointeur sur un objet `YHubPort`, correspondant à le premier port de Yocto-hub accessible en ligne, ou `null` si il n'y a pas de port de Yocto-hub disponibles.

## hubport→describe() YHubPort

Retourne un court texte décrivant le port de Yocto-hub au format TYPE(NAME)=SERIAL.FUNCTIONID.

```
js      function describe( )
nodejs  function describe( )
php     function describe( )
cpp     string describe( )
m       -(NSString*) describe
pas     function describe( ): string
vb      function describe( ) As String
cs      string describe( )
java    String describe( )
py      def describe( )
```

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilsé lors du premier accès a la fonction, SERIAL correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et FUNCTIONID correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La methode va retourner Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1 si le module est déjà connecté ou Relay(BadCustomeName.relay1)=unresolved si le module n'est pas déjà connecté. Cette methode ne declenche aucune transaction USB ou TCP et peut donc être utilisé dans un debuggeur.

**Retourne :**
une chaîne de caractères décrivant le port de Yocto-hub (ex: Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1)

## hubport→get_advertisedValue() YHubPort
## hubport→advertisedValue()

Retourne la valeur courante du port de Yocto-hub (pas plus de 6 caractères).

```
js      function get_advertisedValue( )
nodejs  function get_advertisedValue( )
php     function get_advertisedValue( )
cpp     string get_advertisedValue( )
m       -(NSString*) advertisedValue
pas     function get_advertisedValue( ): string
vb      function get_advertisedValue( ) As String
cs      string get_advertisedValue( )
java    String get_advertisedValue( )
py      def get_advertisedValue( )
cmd     YHubPort target get_advertisedValue
```

**Retourne :**

une chaîne de caractères représentant la valeur courante du port de Yocto-hub (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne `Y_ADVERTISEDVALUE_INVALID`.

## hubport→get_baudRate() YHubPort
## hubport→baudRate()

Retourne la vitesse de transfert utilisée par le port de Yocto-hub, en kbps.

```
js      function get_baudRate( )
nodejs  function get_baudRate( )
php     function get_baudRate( )
cpp     int get_baudRate( )
m       -(int) baudRate
pas     function get_baudRate( ): LongInt
vb      function get_baudRate( ) As Integer
cs      int get_baudRate( )
java    int get_baudRate( )
py      def get_baudRate( )
cmd     YHubPort target get_baudRate
```

La valeur par défaut est 1000 kbps, une valeur inférieure révèle des problèmes de communication.

**Retourne :**
un entier représentant la vitesse de transfert utilisée par le port de Yocto-hub, en kbps

En cas d'erreur, déclenche une exception ou retourne `Y_BAUDRATE_INVALID`.

## hubport→get_enabled() YHubPort
## hubport→enabled()

Retourne vrai si le port du Yocto-hub est alimenté, faux sinon.

```
js      function get_enabled( )
nodejs  function get_enabled( )
php     function get_enabled( )
cpp     Y_ENABLED_enum get_enabled( )
m       -(Y_ENABLED_enum) enabled
pas     function get_enabled( ): Integer
vb      function get_enabled( ) As Integer
cs      int get_enabled( )
java    int get_enabled( )
py      def get_enabled( )
cmd     YHubPort target get_enabled
```

**Retourne :**

soit Y_ENABLED_FALSE, soit Y_ENABLED_TRUE, selon vrai si le port du Yocto-hub est alimenté, faux sinon

En cas d'erreur, déclenche une exception ou retourne Y_ENABLED_INVALID.

## hubport→get_errorMessage() YHubPort
## hubport→errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du port de Yocto-hub.

```
js      function get_errorMessage( )
nodejs  function get_errorMessage( )
php     function get_errorMessage( )
cpp     string get_errorMessage( )
m       -(NSString*) errorMessage
pas     function get_errorMessage( ): string
vb      function get_errorMessage( ) As String
cs      string get_errorMessage( )
java    String get_errorMessage( )
py      def get_errorMessage( )
```

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du port de Yocto-hub.

## hubport→get_errorType() YHubPort
## hubport→errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du port de Yocto-hub.

```
js      function get_errorType( )
nodejs  function get_errorType( )
php     function get_errorType( )
cpp     YRETCODE get_errorType( )
pas     function get_errorType( ): YRETCODE
vb      function get_errorType( ) As YRETCODE
cs      YRETCODE get_errorType( )
java    int get_errorType( )
py      def get_errorType( )
```

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du port de Yocto-hub.

## hubport→get_friendlyName() YHubPort
## hubport→friendlyName()

Retourne un identifiant global du port de Yocto-hub au format NOM_MODULE.NOM_FONCTION.

```
js      function get_friendlyName( )
nodejs  function get_friendlyName( )
php     function get_friendlyName( )
cpp     string get_friendlyName( )
m       -(NSString*) friendlyName
cs      string get_friendlyName( )
java    String get_friendlyName( )
py      def get_friendlyName( )
```

Le chaîne retournée utilise soit les noms logiques du module et du port de Yocto-hub si ils sont définis, soit respectivement le numéro de série du module et l'identifant matériel du port de Yocto-hub (par exemple: MyCustomName.relay1)

**Retourne :**

une chaîne de caractères identifiant le port de Yocto-hub en utilisant les noms logiques (ex: MyCustomName.relay1) En cas d'erreur, déclenche une exception ou retourne Y_FRIENDLYNAME_INVALID.

## hubport→get_functionDescriptor() YHubPort
## hubport→functionDescriptor()

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

```
js      function get_functionDescriptor( )
nodejs  function get_functionDescriptor( )
php     function get_functionDescriptor( )
cpp     YFUN_DESCR get_functionDescriptor( )
m       -(YFUN_DESCR) functionDescriptor
pas     function get_functionDescriptor( ): YFUN_DESCR
vb      function get_functionDescriptor( ) As YFUN_DESCR
cs      YFUN_DESCR get_functionDescriptor( )
java    String get_functionDescriptor( )
py      def get_functionDescriptor( )
```

Cet identifiant peut être utilisé pour tester si deux instance de `YFunction` référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type `YFUN_DESCR`. Si la fonction n'a jamais été contactée, la valeur retournée sera `Y_FUNCTIONDESCRIPTOR_INVALID`

## hubport→get_functionId() YHubPort
## hubport→functionId()

Retourne l'identifiant matériel du port de Yocto-hub, sans référence au module.

```
js      function get_functionId( )
nodejs  function get_functionId( )
php     function get_functionId( )
cpp     string get_functionId( )
m       -(NSString*) functionId
vb      function get_functionId( ) As String
cs      string get_functionId( )
java    String get_functionId( )
py      def get_functionId( )
```

Par example `relay1`.

**Retourne :**

une chaîne de caractères identifiant le port de Yocto-hub (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

## hubport→get_hardwareId() YHubPort
## hubport→hardwareId()

Retourne l'identifiant matériel unique du port de Yocto-hub au format SERIAL.FUNCTIONID.

```
js      function get_hardwareId( )
nodejs  function get_hardwareId( )
php     function get_hardwareId( )
cpp     string get_hardwareId( )
m       -(NSString*) hardwareId
vb      function get_hardwareId( ) As String
cs      string get_hardwareId( )
java    String get_hardwareId( )
py      def get_hardwareId( )
```

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du port de Yocto-hub (par example RELAYLO1-123456.relay1).

**Retourne :**

une chaîne de caractères identifiant le port de Yocto-hub (ex: RELAYLO1-123456.relay1) En cas d'erreur, déclenche une exception ou retourne Y_HARDWAREID_INVALID.

## hubport→get_logicalName() YHubPort
## hubport→logicalName()

Retourne le nom logique du port de Yocto-hub.

```
js      function get_logicalName( )
nodejs  function get_logicalName( )
php     function get_logicalName( )
cpp     string get_logicalName( )
m       -(NSString*) logicalName
pas     function get_logicalName( ): string
vb      function get_logicalName( ) As String
cs      string get_logicalName( )
java    String get_logicalName( )
py      def get_logicalName( )
cmd     YHubPort target get_logicalName
```

**Retourne :**

une chaîne de caractères représentant le nom logique du port de Yocto-hub. En cas d'erreur, déclenche une exception ou retourne `Y_LOGICALNAME_INVALID`.

## hubport→get_module() YHubPort
## hubport→module()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

```
js      function get_module( )
nodejs  function get_module( )
php     function get_module( )
cpp     YModule * get_module( )
m       -(YModule*) module
pas     function get_module( ): TYModule
vb      function get_module( ) As YModule
cs      YModule get_module( )
java    YModule get_module( )
py      def get_module( )
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

**Retourne :**
une instance de YModule

## hubport→get_module_async() YHubPort
## hubport→module_async()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

```
js      function get_module_async( callback, context)
nodejs  function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de YModule

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

## hubport→get_portState() YHubPort
## hubport→portState()

Retourne l'état actuel du port de Yocto-hub.

```
js      function get_portState( )
nodejs  function get_portState( )
php     function get_portState( )
cpp     Y_PORTSTATE_enum get_portState( )
m       -(Y_PORTSTATE_enum) portState
pas     function get_portState( ): Integer
vb      function get_portState( ) As Integer
cs      int get_portState( )
java    int get_portState( )
py      def get_portState( )
cmd     YHubPort target get_portState
```

**Retourne :**

une valeur parmi `Y_PORTSTATE_OFF`, `Y_PORTSTATE_OVRLD`, `Y_PORTSTATE_ON`, `Y_PORTSTATE_RUN` et `Y_PORTSTATE_PROG` représentant l'état actuel du port de Yocto-hub

En cas d'erreur, déclenche une exception ou retourne `Y_PORTSTATE_INVALID`.

## hubport→get_userData() YHubPort
## hubport→userData()

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

| | |
|---|---|
| js | function **get_userData**( ) |
| nodejs | function **get_userData**( ) |
| php | function **get_userData**( ) |
| cpp | void * **get_userData**( ) |
| m | -(void*) userData |
| pas | function **get_userData**( ): Tobject |
| vb | function **get_userData**( ) As Object |
| cs | object **get_userData**( ) |
| java | Object **get_userData**( ) |
| py | def **get_userData**( ) |

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**
l'objet stocké précédemment par l'appelant.

## hubport→isOnline() YHubPort

Vérifie si le module hébergeant le port de Yocto-hub est joignable, sans déclencher d'erreur.

```
js      function isOnline( )
nodejs  function isOnline( )
php     function isOnline( )
cpp     bool isOnline( )
m       -(BOOL) isOnline
pas     function isOnline( ): boolean
vb      function isOnline( ) As Boolean
cs      bool isOnline( )
java    boolean isOnline( )
py      def isOnline( )
```

Si les valeurs des attributs en cache du port de Yocto-hub sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

`true` si le port de Yocto-hub est joignable, `false` sinon

## hubport→isOnline_async() YHubPort

Vérifie si le module hébergeant le port de Yocto-hub est joignable, sans déclencher d'erreur.

```
js      function isOnline_async( callback, context)
nodejs  function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du port de Yocto-hub sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

## hubport→load() YHubPort

Met en cache les valeurs courantes du port de Yocto-hub, avec une durée de validité spécifiée.

```
js      function load( msValidity)
nodejs  function load( msValidity)
php     function load( $msValidity)
cpp     YRETCODE load( int msValidity)
m       -(YRETCODE) load : (int) msValidity
pas     function load( msValidity: integer): YRETCODE
vb      function load( ByVal msValidity As Integer) As YRETCODE
cs      YRETCODE load( int msValidity)
java    int load( long msValidity)
py      def load( msValidity)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## hubport→load_async() YHubPort

Met en cache les valeurs courantes du port de Yocto-hub, avec une durée de validité spécifiée.

```
js      function load_async( msValidity, callback, context)
nodejs  function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

| | |
|---|---|
| **msValidity** | un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes |
| **callback** | fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou YAPI_SUCCESS) |
| **context** | contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback |

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

## hubport→nextHubPort() YHubPort

Continue l'énumération des port de Yocto-hub commencée à l'aide de `yFirstHubPort()`.

```
js      function nextHubPort( )
nodejs  function nextHubPort( )
php     function nextHubPort( )
cpp     YHubPort * nextHubPort( )
m       -(YHubPort*) nextHubPort
pas     function nextHubPort( ): TYHubPort
vb      function nextHubPort( ) As YHubPort
cs      YHubPort nextHubPort( )
java    YHubPort nextHubPort( )
py      def nextHubPort( )
```

**Retourne :**

un pointeur sur un objet `YHubPort` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## hubport→registerValueCallback() YHubPort

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

| | |
|---|---|
| js | function **registerValueCallback**( **callback**) |
| nodejs | function **registerValueCallback**( **callback**) |
| php | function **registerValueCallback**( **$callback**) |
| cpp | int **registerValueCallback**( YHubPortValueCallback **callback**) |
| m | -(int) **registerValueCallback** : (YHubPortValueCallback) **callback** |
| pas | function **registerValueCallback**( **callback**: TYHubPortValueCallback): LongInt |
| vb | function **registerValueCallback**( ) As Integer |
| cs | int **registerValueCallback**( ValueCallback **callback**) |
| java | int **registerValueCallback**( UpdateCallback **callback**) |
| py | def **registerValueCallback**( **callback**) |

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

**Paramètres :**

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

## hubport→set_enabled() YHubPort
## hubport→setEnabled()

Modifie le mode d'activation du port du Yocto-hub.

```
js      function set_enabled( newval)
nodejs  function set_enabled( newval)
php     function set_enabled( $newval)
cpp     int set_enabled( Y_ENABLED_enum newval)
m       -(int) setEnabled : (Y_ENABLED_enum) newval
pas     function set_enabled( newval: Integer): integer
vb      function set_enabled( ByVal newval As Integer) As Integer
cs      int set_enabled( int newval)
java    int set_enabled( int newval)
py      def set_enabled( newval)
cmd     YHubPort target set_enabled newval
```

Si le port est actif, il sera alimenté. Sinon, l'alimentation du module est coupée.

**Paramètres :**

**newval** soit Y_ENABLED_FALSE, soit Y_ENABLED_TRUE, selon le mode d'activation du port du Yocto-hub

**Retourne :**

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## hubport→set_logicalName() YHubPort
## hubport→setLogicalName()

Modifie le nom logique du port de Yocto-hub.

```
js      function set_logicalName( newval)
nodejs  function set_logicalName( newval)
php     function set_logicalName( $newval)
cpp     int set_logicalName( const string& newval)
m       -(int) setLogicalName : (NSString*) newval
pas     function set_logicalName( newval: string): integer
vb      function set_logicalName( ByVal newval As String) As Integer
cs      int set_logicalName( string newval)
java    int set_logicalName( String newval)
py      def set_logicalName( newval)
cmd     YHubPort target set_logicalName newval
```

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique du port de Yocto-hub.

**Retourne :**

`YAPI_SUCCESS` si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## hubport→set_userData() YHubPort
## hubport→setUserData()

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

```
js      function set_userData( data)
nodejs  function set_userData( data)
php     function set_userData( $data)
cpp     void set_userData( void* data)
m       -(void) setUserData : (void*) data
pas     procedure set_userData( data: Tobject)
vb      procedure set_userData( ByVal data As Object)
cs      void set_userData( object data)
java    void set_userData( Object data)
py      def set_userData( data)
```

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

## hubport→wait_async() YHubPort

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js      function wait_async( callback, context)
nodejs  function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

# 10.2. Interface de la fonction Network

Les objets YNetwork permettent de contrôler les paramètres TCP/IP des modules Yoctopuce dotés d'une interface réseau.

Pour utiliser les fonctions décrites ici, vous devez inclure:

```
js      <script type='text/javascript' src='yocto_network.js'></script>
nodejs  var yoctolib = require('yoctolib');
        var YNetwork = yoctolib.YNetwork;
php     require_once('yocto_network.php');
cpp     #include "yocto_network.h"
m       #import "yocto_network.h"
pas     uses yocto_network;
vb      yocto_network.vb
cs      yocto_network.cs
java    import com.yoctopuce.YoctoAPI.YNetwork;
py      from yocto_network import *
```

| Fonction globales |
|---|
| **yFindNetwork**(**func**)<br>Permet de retrouver une interface réseau d'après un identifiant donné. |
| **yFirstNetwork**()<br>Commence l'énumération des interfaces réseau accessibles par la librairie. |
| **Méthodes des objets `YNetwork`** |
| **network→callbackLogin**(**username**, **password**)<br>Contacte le callback de notification et sauvegarde un laisser-passer pour s'y connecter. |
| **network→describe**()<br>Retourne un court texte décrivant l'interface réseau au format `TYPE(NAME)=SERIAL.FUNCTIONID`. |
| **network→get_adminPassword**()<br>Retourne une chaîne de hash si un mot de passe a été configuré pour l'utilisateur "admin", ou sinon une chaîne vide. |
| **network→get_advertisedValue**()<br>Retourne la valeur courante de l'interface réseau (pas plus de 6 caractères). |
| **network→get_callbackCredentials**()<br>Retourne une version hashée du laisser-passer pour le callback de notification s'il a été configuré, ou sinon une chaîne vide. |
| **network→get_callbackEncoding**()<br>Retourne l'encodage à utiliser pour représenter les valeurs notifiées par callback. |
| **network→get_callbackMaxDelay**()<br>Retourne l'attente maximale entre deux notifications par callback, en secondes. |
| **network→get_callbackMethod**()<br>Retourne la méthode HTTP à utiliser pour signaler les changements d'état par callback. |
| **network→get_callbackMinDelay**()<br>Retourne l'attente minimale entre deux notifications par callback, en secondes. |
| **network→get_callbackUrl**()<br>Retourne l'adresse (URL) de callback à notifier lors de changement d'état significatifs. |
| **network→get_discoverable**()<br>Retourne l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocols uPnP/Bonjour). |

**network→get_errorMessage**()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'interface réseau.

**network→get_errorType**()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'interface réseau.

**network→get_friendlyName**()

Retourne un identifiant global de l'interface réseau au format `NOM_MODULE.NOM_FONCTION`.

**network→get_functionDescriptor**()

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

**network→get_functionId**()

Retourne l'identifiant matériel de l'interface réseau, sans référence au module.

**network→get_hardwareId**()

Retourne l'identifiant matériel unique de l'interface réseau au format `SERIAL.FUNCTIONID`.

**network→get_ipAddress**()

Retourne l'adresse IP utilisée par le module Yoctopuce.

**network→get_logicalName**()

Retourne le nom logique de l'interface réseau.

**network→get_macAddress**()

Retourne l'adresse MAC de l'interface réseau, unique pour chaque module.

**network→get_module**()

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**network→get_module_async**(**callback**, **context**)

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**network→get_poeCurrent**()

Retourne le courant consommé par le module depuis Power-over-Ethernet (PoE), en milliampères.

**network→get_primaryDNS**()

Retourne l'adresse IP du serveur de noms primaire que le module doit utiliser.

**network→get_readiness**()

Retourne l'état de fonctionnement atteint par l'interface réseau.

**network→get_router**()

Retourne l'adresse IP du routeur (passerelle) utilisé par le module (*default gateway*).

**network→get_secondaryDNS**()

Retourne l'adresse IP du serveur de noms secondaire que le module doit utiliser.

**network→get_subnetMask**()

Retourne le masque de sous-réseau utilisé par le module.

**network→get_userData**()

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

**network→get_userPassword**()

Retourne une chaîne de hash si un mot de passe a été configuré pour l'utilisateur "user", ou sinon une chaîne vide.

**network→get_wwwWatchdogDelay**()

Retourne la durée de perte de connection WWW tolérée (en secondes) avant de déclancher un redémarrage automatique pour tenter de récupérer la connectivité Internet.

**network→isOnline**()

Vérifie si le module hébergeant l'interface réseau est joignable, sans déclencher d'erreur.

**network→isOnline_async**(**callback**, **context**)

Vérifie si le module hébergeant l'interface réseau est joignable, sans déclencher d'erreur.

**network→load(msValidity)**

Met en cache les valeurs courantes de l'interface réseau, avec une durée de validité spécifiée.

**network→load_async(msValidity, callback, context)**

Met en cache les valeurs courantes de l'interface réseau, avec une durée de validité spécifiée.

**network→nextNetwork()**

Continue l'énumération des interfaces réseau commencée à l'aide de `yFirstNetwork()`.

**network→ping(host)**

Ping str_host pour vérifier la connexion réseau.

**network→registerValueCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

**network→set_adminPassword(newval)**

Modifie le mot de passe pour l'utilisateur "admin", qui devient alors instantanément nécessaire pour toute altération de l'état du module.

**network→set_callbackCredentials(newval)**

Modifie le laisser-passer pour se connecter à l'adresse de callback.

**network→set_callbackEncoding(newval)**

Modifie l'encodage à utiliser pour représenter les valeurs notifiées par callback.

**network→set_callbackMaxDelay(newval)**

Modifie l'attente maximale entre deux notifications par callback, en secondes.

**network→set_callbackMethod(newval)**

Modifie la méthode HTTP à utiliser pour signaler les changements d'état par callback.

**network→set_callbackMinDelay(newval)**

Modifie l'attente minimale entre deux notifications par callback, en secondes.

**network→set_callbackUrl(newval)**

Modifie l'adresse (URL) de callback à notifier lors de changement d'état significatifs.

**network→set_discoverable(newval)**

Modifie l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocols uPnP/Bonjour).

**network→set_logicalName(newval)**

Modifie le nom logique de l'interface réseau.

**network→set_primaryDNS(newval)**

Modifie l'adresse IP du serveur de noms primaire que le module doit utiliser.

**network→set_secondaryDNS(newval)**

Modifie l'adresse IP du serveur de nom secondaire que le module doit utiliser.

**network→set_userData(data)**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

**network→set_userPassword(newval)**

Modifie le mode de passe pour l'utilisateur "user", qui devient alors instantanément nécessaire pour tout accès au module.

**network→set_wwwWatchdogDelay(newval)**

Modifie la durée de perte de connection WWW tolérée (en secondes) avant de déclancher un redémarrage automatique pour tenter de récupérer la connectivité Internet.

**network→useDHCP(fallbackIpAddr, fallbackSubnetMaskLen, fallbackRouter)**

Modifie la configuration de l'interface réseau pour utiliser une adresse assignée automatiquement par le serveur DHCP.

**network→useStaticIP(ipAddress, subnetMaskLen, router)**

Modifie la configuration de l'interface réseau pour utiliser une adresse IP assignée manuellement (adresse IP statique).

**network→wait_async(callback, context)**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YNetwork.FindNetwork() YNetwork
## yFindNetwork()

Permet de retrouver une interface réseau d'après un identifiant donné.

```
js      function yFindNetwork( func)
nodejs  function FindNetwork( func)
php     function yFindNetwork( $func)
cpp     YNetwork* yFindNetwork( const string& func)
m       YNetwork* yFindNetwork( NSString* func)
pas     function yFindNetwork( func: string): TYNetwork
vb      function yFindNetwork( ByVal func As String) As YNetwork
cs      YNetwork FindNetwork( string func)
java    YNetwork FindNetwork( String func)
py      def FindNetwork( func)
```

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que l'interface réseau soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YNetwork.isOnline()` pour tester si l'interface réseau est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

**Paramètres :**

**func** une chaîne de caractères qui référence l'interface réseau sans ambiguïté

**Retourne :**

un objet de classe `YNetwork` qui permet ensuite de contrôler l'interface réseau.

## YNetwork.FirstNetwork() YNetwork
## yFirstNetwork()

Commence l'énumération des interfaces réseau accessibles par la librairie.

```
js      function yFirstNetwork( )
nodejs  function FirstNetwork( )
php     function yFirstNetwork( )
cpp     YNetwork* yFirstNetwork( )
m       YNetwork* yFirstNetwork( )
pas     function yFirstNetwork( ): TYNetwork
vb      function yFirstNetwork( ) As YNetwork
cs      YNetwork FirstNetwork( )
java    YNetwork FirstNetwork( )
py      def FirstNetwork( )
```

Utiliser la fonction `YNetwork.nextNetwork()` pour itérer sur les autres interfaces réseau.

**Retourne :**

un pointeur sur un objet `YNetwork`, correspondant à la première interface réseau accessible en ligne, ou `null` si il n'y a pas de interfaces réseau disponibles.

## network→callbackLogin() YNetwork

Contacte le callback de notification et sauvegarde un laisser-passer pour s'y connecter.

```
js      function callbackLogin( username, password)
nodejs  function callbackLogin( username, password)
php     function callbackLogin( $username, $password)
cpp     int callbackLogin( string username, string password)
m       -(int) callbackLogin : (NSString*) username : (NSString*) password
pas     function callbackLogin( username: string, password: string): integer
vb      function callbackLogin( ByVal username As String,
                                 ByVal password As String) As Integer
cs      int callbackLogin( string username, string password)
java    int callbackLogin( String username, String password)
py      def callbackLogin( username, password)
cmd     YNetwork target callbackLogin username password
```

Le mot de passe ne sera pas stocké dans le module, mais seulement une version hashée non réversible. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**username** nom d'utilisateur pour s'identifier au callback

**password** mot de passe pour s'identifier au callback

**Retourne :**

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## network→describe() YNetwork

Retourne un court texte décrivant l'interface réseau au format TYPE(NAME)=SERIAL.FUNCTIONID.

```
js      function describe( )
nodejs  function describe( )
php     function describe( )
cpp     string describe( )
m       -(NSString*) describe
pas     function describe( ): string
vb      function describe( ) As String
cs      string describe( )
java    String describe( )
py      def describe( )
```

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilsé lors du premier accès a la fonction, SERIAL correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et FUNCTIONID correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La methode va retourner Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1 si le module est déjà connecté ou Relay(BadCustomeName.relay1)=unresolved si le module n'est pas déjà connecté. Cette methode ne declenche aucune transaction USB ou TCP et peut donc être utilisé dans un debuggeur.

**Retourne :**
une chaîne de caractères décrivant l'interface réseau (ex: Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1)

## network→get_adminPassword() YNetwork
## network→adminPassword()

Retourne une chaîne de hash si un mot de passe a été configuré pour l'utilisateur "admin", ou sinon une chaîne vide.

```
js      function get_adminPassword( )
nodejs  function get_adminPassword( )
php     function get_adminPassword( )
cpp     string get_adminPassword( )
m       -(NSString*) adminPassword
pas     function get_adminPassword( ): string
vb      function get_adminPassword( ) As String
cs      string get_adminPassword( )
java    String get_adminPassword( )
py      def get_adminPassword( )
cmd     YNetwork target get_adminPassword
```

**Retourne :**

une chaîne de caractères représentant une chaîne de hash si un mot de passe a été configuré pour l'utilisateur "admin", ou sinon une chaîne vide

En cas d'erreur, déclenche une exception ou retourne `Y_ADMINPASSWORD_INVALID`.

## network→get_advertisedValue() YNetwork
## network→advertisedValue()

Retourne la valeur courante de l'interface réseau (pas plus de 6 caractères).

```
js      function get_advertisedValue( )
nodejs  function get_advertisedValue( )
php     function get_advertisedValue( )
cpp     string get_advertisedValue( )
m       -(NSString*) advertisedValue
pas     function get_advertisedValue( ): string
vb      function get_advertisedValue( ) As String
cs      string get_advertisedValue( )
java    String get_advertisedValue( )
py      def get_advertisedValue( )
cmd     YNetwork target get_advertisedValue
```

**Retourne :**

une chaîne de caractères représentant la valeur courante de l'interface réseau (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne `Y_ADVERTISEDVALUE_INVALID`.

## network→get_callbackCredentials() YNetwork
## network→callbackCredentials()

Retourne une version hashée du laisser-passer pour le callback de notification s'il a été configuré, ou sinon une chaîne vide.

```
js      function get_callbackCredentials( )
nodejs  function get_callbackCredentials( )
php     function get_callbackCredentials( )
cpp     string get_callbackCredentials( )
m       -(NSString*) callbackCredentials
pas     function get_callbackCredentials( ): string
vb      function get_callbackCredentials( ) As String
cs      string get_callbackCredentials( )
java    String get_callbackCredentials( )
py      def get_callbackCredentials( )
cmd     YNetwork target get_callbackCredentials
```

**Retourne :**

une chaîne de caractères représentant une version hashée du laisser-passer pour le callback de notification s'il a été configuré, ou sinon une chaîne vide

En cas d'erreur, déclenche une exception ou retourne `Y_CALLBACKCREDENTIALS_INVALID`.

## network→get_callbackEncoding() YNetwork
## network→callbackEncoding()

Retourne l'encodage à utiliser pour représenter les valeurs notifiées par callback.

| | |
|---|---|
| js | function **get_callbackEncoding**( ) |
| nodejs | function **get_callbackEncoding**( ) |
| php | function **get_callbackEncoding**( ) |
| cpp | Y_CALLBACKENCODING_enum **get_callbackEncoding**( ) |
| m | -(Y_CALLBACKENCODING_enum) callbackEncoding |
| pas | function **get_callbackEncoding**( ): Integer |
| vb | function **get_callbackEncoding**( ) As Integer |
| cs | int **get_callbackEncoding**( ) |
| java | int **get_callbackEncoding**( ) |
| py | def **get_callbackEncoding**( ) |
| cmd | YNetwork **target get_callbackEncoding** |

**Retourne :**

une valeur parmi `Y_CALLBACKENCODING_FORM`, `Y_CALLBACKENCODING_JSON`, `Y_CALLBACKENCODING_JSON_ARRAY`, `Y_CALLBACKENCODING_CSV` et `Y_CALLBACKENCODING_YOCTO_API` représentant l'encodage à utiliser pour représenter les valeurs notifiées par callback

En cas d'erreur, déclenche une exception ou retourne `Y_CALLBACKENCODING_INVALID`.

## network→get_callbackMaxDelay() YNetwork
## network→callbackMaxDelay()

Retourne l'attente maximale entre deux notifications par callback, en secondes.

```
js      function get_callbackMaxDelay( )
nodejs  function get_callbackMaxDelay( )
php     function get_callbackMaxDelay( )
cpp     int get_callbackMaxDelay( )
m       -(int) callbackMaxDelay
pas     function get_callbackMaxDelay( ): LongInt
vb      function get_callbackMaxDelay( ) As Integer
cs      int get_callbackMaxDelay( )
java    int get_callbackMaxDelay( )
py      def get_callbackMaxDelay( )
cmd     YNetwork target get_callbackMaxDelay
```

**Retourne :**

un entier représentant l'attente maximale entre deux notifications par callback, en secondes

En cas d'erreur, déclenche une exception ou retourne `Y_CALLBACKMAXDELAY_INVALID`.

## network→get_callbackMethod() YNetwork
## network→callbackMethod()

Retourne la méthode HTTP à utiliser pour signaler les changements d'état par callback.

```
js      function get_callbackMethod( )
nodejs  function get_callbackMethod( )
php     function get_callbackMethod( )
cpp     Y_CALLBACKMETHOD_enum get_callbackMethod( )
m       -(Y_CALLBACKMETHOD_enum) callbackMethod
pas     function get_callbackMethod( ): Integer
vb      function get_callbackMethod( ) As Integer
cs      int get_callbackMethod( )
java    int get_callbackMethod( )
py      def get_callbackMethod( )
cmd     YNetwork target get_callbackMethod
```

**Retourne :**

une valeur parmi Y_CALLBACKMETHOD_POST, Y_CALLBACKMETHOD_GET et Y_CALLBACKMETHOD_PUT représentant la méthode HTTP à utiliser pour signaler les changements d'état par callback

En cas d'erreur, déclenche une exception ou retourne Y_CALLBACKMETHOD_INVALID.

## network→get_callbackMinDelay() YNetwork
## network→callbackMinDelay()

Retourne l'attente minimale entre deux notifications par callback, en secondes.

```
js      function get_callbackMinDelay( )
nodejs  function get_callbackMinDelay( )
php     function get_callbackMinDelay( )
cpp     int get_callbackMinDelay( )
m       -(int) callbackMinDelay
pas     function get_callbackMinDelay( ): LongInt
vb      function get_callbackMinDelay( ) As Integer
cs      int get_callbackMinDelay( )
java    int get_callbackMinDelay( )
py      def get_callbackMinDelay( )
cmd     YNetwork target get_callbackMinDelay
```

**Retourne :**

un entier représentant l'attente minimale entre deux notifications par callback, en secondes

En cas d'erreur, déclenche une exception ou retourne `Y_CALLBACKMINDELAY_INVALID`.

## network→get_callbackUrl() YNetwork
## network→callbackUrl()

Retourne l'adresse (URL) de callback à notifier lors de changement d'état significatifs.

```
js      function get_callbackUrl( )
nodejs  function get_callbackUrl( )
php     function get_callbackUrl( )
cpp     string get_callbackUrl( )
m       -(NSString*) callbackUrl
pas     function get_callbackUrl( ): string
vb      function get_callbackUrl( ) As String
cs      string get_callbackUrl( )
java    String get_callbackUrl( )
py      def get_callbackUrl( )
cmd     YNetwork target get_callbackUrl
```

**Retourne :**

une chaîne de caractères représentant l'adresse (URL) de callback à notifier lors de changement d'état significatifs

En cas d'erreur, déclenche une exception ou retourne `Y_CALLBACKURL_INVALID`.

## network→get_discoverable() YNetwork
## network→discoverable()

Retourne l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocols uPnP/Bonjour).

```
js      function get_discoverable( )
nodejs  function get_discoverable( )
php     function get_discoverable( )
cpp     Y_DISCOVERABLE_enum get_discoverable( )
m       -(Y_DISCOVERABLE_enum) discoverable
pas     function get_discoverable( ): Integer
vb      function get_discoverable( ) As Integer
cs      int get_discoverable( )
java    int get_discoverable( )
py      def get_discoverable( )
cmd     YNetwork target get_discoverable
```

**Retourne :**

soit `Y_DISCOVERABLE_FALSE`, soit `Y_DISCOVERABLE_TRUE`, selon l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocols uPnP/Bonjour)

En cas d'erreur, déclenche une exception ou retourne `Y_DISCOVERABLE_INVALID`.

## network→get_errorMessage() YNetwork
## network→errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'interface réseau.

```
js      function get_errorMessage( )
nodejs  function get_errorMessage( )
php     function get_errorMessage( )
cpp     string get_errorMessage( )
m       -(NSString*) errorMessage
pas     function get_errorMessage( ): string
vb      function get_errorMessage( ) As String
cs      string get_errorMessage( )
java    String get_errorMessage( )
py      def get_errorMessage( )
```

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation de l'interface réseau.

## network→get_errorType() YNetwork
## network→errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'interface réseau.

```
js      function get_errorType( )
nodejs  function get_errorType( )
php     function get_errorType( )
cpp     YRETCODE get_errorType( )
pas     function get_errorType( ): YRETCODE
vb      function get_errorType( ) As YRETCODE
cs      YRETCODE get_errorType( )
java    int get_errorType( )
py      def get_errorType( )
```

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**
un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation de l'interface réseau.

## network→get_friendlyName() YNetwork
## network→friendlyName()

Retourne un identifiant global de l'interface réseau au format NOM_MODULE.NOM_FONCTION.

```
js      function get_friendlyName( )
nodejs  function get_friendlyName( )
php     function get_friendlyName( )
cpp     string get_friendlyName( )
m       -(NSString*) friendlyName
cs      string get_friendlyName( )
java    String get_friendlyName( )
py      def get_friendlyName( )
```

Le chaîne retournée utilise soit les noms logiques du module et de l'interface réseau si ils sont définis, soit respectivement le numéro de série du module et l'identifant matériel de l'interface réseau (par exemple: MyCustomName.relay1)

**Retourne :**

une chaîne de caractères identifiant l'interface réseau en utilisant les noms logiques (ex: MyCustomName.relay1) En cas d'erreur, déclenche une exception ou retourne Y_FRIENDLYNAME_INVALID.

## network→get_functionDescriptor() YNetwork
## network→functionDescriptor()

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

```
js      function get_functionDescriptor( )
nodejs  function get_functionDescriptor( )
php     function get_functionDescriptor( )
cpp     YFUN_DESCR get_functionDescriptor( )
m       -(YFUN_DESCR) functionDescriptor
pas     function get_functionDescriptor( ): YFUN_DESCR
vb      function get_functionDescriptor( ) As YFUN_DESCR
cs      YFUN_DESCR get_functionDescriptor( )
java    String get_functionDescriptor( )
py      def get_functionDescriptor( )
```

Cet identifiant peut être utilisé pour tester si deux instance de `YFunction` référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type `YFUN_DESCR`. Si la fonction n'a jamais été contactée, la valeur retournée sera `Y_FUNCTIONDESCRIPTOR_INVALID`

## network→get_functionId() YNetwork
## network→functionId()

Retourne l'identifiant matériel de l'interface réseau, sans référence au module.

```
js      function get_functionId( )
nodejs  function get_functionId( )
php     function get_functionId( )
cpp     string get_functionId( )
m       -(NSString*) functionId
vb      function get_functionId( ) As String
cs      string get_functionId( )
java    String get_functionId( )
py      def get_functionId( )
```

Par example `relay1`.

**Retourne :**

une chaîne de caractères identifiant l'interface réseau (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

## network→get_hardwareId() YNetwork
## network→hardwareId()

Retourne l'identifiant matériel unique de l'interface réseau au format SERIAL.FUNCTIONID.

```
js      function get_hardwareId( )
nodejs  function get_hardwareId( )
php     function get_hardwareId( )
cpp     string get_hardwareId( )
m       -(NSString*) hardwareId
vb      function get_hardwareId( ) As String
cs      string get_hardwareId( )
java    String get_hardwareId( )
py      def get_hardwareId( )
```

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de l'interface réseau (par example RELAYLO1-123456.relay1).

**Retourne :**

une chaîne de caractères identifiant l'interface réseau (ex: RELAYLO1-123456.relay1) En cas d'erreur, déclenche une exception ou retourne Y_HARDWAREID_INVALID.

## network→get_ipAddress() YNetwork
## network→ipAddress()

Retourne l'adresse IP utilisée par le module Yoctopuce.

```
js      function get_ipAddress( )
nodejs  function get_ipAddress( )
php     function get_ipAddress( )
cpp     string get_ipAddress( )
m       -(NSString*) ipAddress
pas     function get_ipAddress( ): string
vb      function get_ipAddress( ) As String
cs      string get_ipAddress( )
java    String get_ipAddress( )
py      def get_ipAddress( )
cmd     YNetwork target get_ipAddress
```

Il peut s'agir d'une adresse configurée statiquement, ou d'une adresse reçue par un serveur DHCP.

**Retourne :**
une chaîne de caractères représentant l'adresse IP utilisée par le module Yoctopuce

En cas d'erreur, déclenche une exception ou retourne `Y_IPADDRESS_INVALID`.

## network→get_logicalName() YNetwork
## network→logicalName()

Retourne le nom logique de l'interface réseau.

```
js      function get_logicalName( )
nodejs  function get_logicalName( )
php     function get_logicalName( )
cpp     string get_logicalName( )
m       -(NSString*) logicalName
pas     function get_logicalName( ): string
vb      function get_logicalName( ) As String
cs      string get_logicalName( )
java    String get_logicalName( )
py      def get_logicalName( )
cmd     YNetwork target get_logicalName
```

**Retourne :**

une chaîne de caractères représentant le nom logique de l'interface réseau. En cas d'erreur, déclenche une exception ou retourne `Y_LOGICALNAME_INVALID`.

## network→get_macAddress() YNetwork
## network→macAddress()

Retourne l'adresse MAC de l'interface réseau, unique pour chaque module.

```
js      function get_macAddress( )
nodejs  function get_macAddress( )
php     function get_macAddress( )
cpp     string get_macAddress( )
m       -(NSString*) macAddress
pas     function get_macAddress( ): string
vb      function get_macAddress( ) As String
cs      string get_macAddress( )
java    String get_macAddress( )
py      def get_macAddress( )
cmd     YNetwork target get_macAddress
```

L'adresse MAC est aussi présente sur un autocollant sur le module, représentée en chiffres et en code-barres.

**Retourne :**
une chaîne de caractères représentant l'adresse MAC de l'interface réseau, unique pour chaque module

En cas d'erreur, déclenche une exception ou retourne `Y_MACADDRESS_INVALID`.

## network→get_module() YNetwork
## network→module()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

```
js      function get_module( )
nodejs  function get_module( )
php     function get_module( )
cpp     YModule * get_module( )
m       -(YModule*) module
pas     function get_module( ): TYModule
vb      function get_module( ) As YModule
cs      YModule get_module( )
java    YModule get_module( )
py      def get_module( )
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

**Retourne :**
une instance de YModule

## network→get_module_async() YNetwork
## network→module_async()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

```
js      function get_module_async( callback, context)
nodejs  function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de YModule

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

## network→get_poeCurrent() YNetwork
## network→poeCurrent()

Retourne le courant consommé par le module depuis Power-over-Ethernet (PoE), en milliampères.

```
js      function get_poeCurrent( )
nodejs  function get_poeCurrent( )
php     function get_poeCurrent( )
cpp     int get_poeCurrent( )
m       -(int) poeCurrent
pas     function get_poeCurrent( ): LongInt
vb      function get_poeCurrent( ) As Integer
cs      int get_poeCurrent( )
java    int get_poeCurrent( )
py      def get_poeCurrent( )
cmd     YNetwork target get_poeCurrent
```

La consommation est mesurée après conversion en 5 Volt, et ne doit jamais dépasser 1800 mA.

**Retourne :**

un entier représentant le courant consommé par le module depuis Power-over-Ethernet (PoE), en milliampères

En cas d'erreur, déclenche une exception ou retourne `Y_POECURRENT_INVALID`.

## network→get_primaryDNS() YNetwork
## network→primaryDNS()

Retourne l'adresse IP du serveur de noms primaire que le module doit utiliser.

```
js      function get_primaryDNS( )
nodejs  function get_primaryDNS( )
php     function get_primaryDNS( )
cpp     string get_primaryDNS( )
m       -(NSString*) primaryDNS
pas     function get_primaryDNS( ): string
vb      function get_primaryDNS( ) As String
cs      string get_primaryDNS( )
java    String get_primaryDNS( )
py      def get_primaryDNS( )
cmd     YNetwork target get_primaryDNS
```

**Retourne :**
une chaîne de caractères représentant l'adresse IP du serveur de noms primaire que le module doit utiliser

En cas d'erreur, déclenche une exception ou retourne `Y_PRIMARYDNS_INVALID`.

## network→get_readiness() YNetwork
## network→readiness()

Retourne l'état de fonctionnement atteint par l'interface réseau.

```
js      function get_readiness( )
nodejs  function get_readiness( )
php     function get_readiness( )
cpp     Y_READINESS_enum get_readiness( )
m       -(Y_READINESS_enum) readiness
pas     function get_readiness( ): Integer
vb      function get_readiness( ) As Integer
cs      int get_readiness( )
java    int get_readiness( )
py      def get_readiness( )
cmd     YNetwork target get_readiness
```

Le niveau zéro (DOWN_0) signifie qu'aucun support réseau matériel n'a été détecté. Soit il n'y a pas de signal sur le cable réseau, soit le point d'accès sans fil choisi n'est pas détecté. Le niveau 1 (LIVE_1) est atteint lorsque le réseau est détecté, mais n'est pas encore connecté. Pour un réseau sans fil, cela confirme la l'existence du SSID configuré. Le niveau 2 (LINK_2) est atteint lorsque le support matériel du réseau est fonctionnel. Pour une connection réseau filaire, le niveau 2 signifie que le cable est connecté aux deux bouts. Pour une connection à un point d'accès réseau sans fil, il démontre que les paramètres de sécurités configurés sont corrects. Pour une connection sans fil en mode ad-hoc, cela signifie qu'il y a au moins un partenaire sur le réseau ad-hoc. Le niveau 3 (DHCP_3) est atteint lorsque qu'une adresse IP a été obtenue par DHCP. Le niveau 4 (DNS_4) est atteint lorsqu'un serveur DNS est joignable par le réseau. Le niveau 5 (WWW_5) est atteint lorsque la connectivité globale à internet est avérée par l'obtention de l'heure courante sur une serveur NTP.

**Retourne :**

une valeur parmi `Y_READINESS_DOWN`, `Y_READINESS_EXISTS`, `Y_READINESS_LINKED`, `Y_READINESS_LAN_OK` et `Y_READINESS_WWW_OK` représentant l'état de fonctionnement atteint par l'interface réseau

En cas d'erreur, déclenche une exception ou retourne `Y_READINESS_INVALID`.

## network→get_router() YNetwork
## network→router()

Retourne l'adresse IP du routeur (passerelle) utilisé par le module (*default gateway*).

```
js      function get_router( )
nodejs  function get_router( )
php     function get_router( )
cpp     string get_router( )
m       -(NSString*) router
pas     function get_router( ): string
vb      function get_router( ) As String
cs      string get_router( )
java    String get_router( )
py      def get_router( )
cmd     YNetwork target get_router
```

**Retourne :**

une chaîne de caractères représentant l'adresse IP du routeur (passerelle) utilisé par le module (*default gateway*)

En cas d'erreur, déclenche une exception ou retourne `Y_ROUTER_INVALID`.

## network→get_secondaryDNS() YNetwork
## network→secondaryDNS()

Retourne l'adresse IP du serveur de noms secondaire que le module doit utiliser.

```
js      function get_secondaryDNS( )
nodejs  function get_secondaryDNS( )
php     function get_secondaryDNS( )
cpp     string get_secondaryDNS( )
m       -(NSString*) secondaryDNS
pas     function get_secondaryDNS( ): string
vb      function get_secondaryDNS( ) As String
cs      string get_secondaryDNS( )
java    String get_secondaryDNS( )
py      def get_secondaryDNS( )
cmd     YNetwork target get_secondaryDNS
```

**Retourne :**

une chaîne de caractères représentant l'adresse IP du serveur de noms secondaire que le module doit utiliser

En cas d'erreur, déclenche une exception ou retourne `Y_SECONDARYDNS_INVALID`.

## network→get_subnetMask() YNetwork
## network→subnetMask()

Retourne le masque de sous-réseau utilisé par le module.

```
js      function get_subnetMask( )
nodejs  function get_subnetMask( )
php     function get_subnetMask( )
cpp     string get_subnetMask( )
m       -(NSString*) subnetMask
pas     function get_subnetMask( ): string
vb      function get_subnetMask( ) As String
cs      string get_subnetMask( )
java    String get_subnetMask( )
py      def get_subnetMask( )
cmd     YNetwork target get_subnetMask
```

**Retourne :**
une chaîne de caractères représentant le masque de sous-réseau utilisé par le module

En cas d'erreur, déclenche une exception ou retourne `Y_SUBNETMASK_INVALID`.

## network→get_userData() YNetwork
## network→userData()

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

```
js      function get_userData( )
nodejs  function get_userData( )
php     function get_userData( )
cpp     void * get_userData( )
m       -(void*) userData
pas     function get_userData( ): Tobject
vb      function get_userData( ) As Object
cs      object get_userData( )
java    Object get_userData( )
py      def get_userData( )
```

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**
l'objet stocké précédemment par l'appelant.

## network→get_userPassword() YNetwork
## network→userPassword()

Retourne une chaîne de hash si un mot de passe a été configuré pour l'utilisateur "user", ou sinon une chaîne vide.

```
js      function get_userPassword( )
nodejs  function get_userPassword( )
php     function get_userPassword( )
cpp     string get_userPassword( )
m       -(NSString*) userPassword
pas     function get_userPassword( ): string
vb      function get_userPassword( ) As String
cs      string get_userPassword( )
java    String get_userPassword( )
py      def get_userPassword( )
cmd     YNetwork target get_userPassword
```

**Retourne :**

une chaîne de caractères représentant une chaîne de hash si un mot de passe a été configuré pour l'utilisateur "user", ou sinon une chaîne vide

En cas d'erreur, déclenche une exception ou retourne `Y_USERPASSWORD_INVALID`.

## network→get_wwwWatchdogDelay() YNetwork
## network→wwwWatchdogDelay()

Retourne la durée de perte de connection WWW tolérée (en secondes) avant de déclancher un redémarrage automatique pour tenter de récupérer la connectivité Internet.

```
js      function get_wwwWatchdogDelay( )
nodejs  function get_wwwWatchdogDelay( )
php     function get_wwwWatchdogDelay( )
cpp     int get_wwwWatchdogDelay( )
m       -(int) wwwWatchdogDelay
pas     function get_wwwWatchdogDelay( ): LongInt
vb      function get_wwwWatchdogDelay( ) As Integer
cs      int get_wwwWatchdogDelay( )
java    int get_wwwWatchdogDelay( )
py      def get_wwwWatchdogDelay( )
cmd     YNetwork target get_wwwWatchdogDelay
```

Une valeur nulle désactive le redémarrage automatique en cas de perte de connectivité WWW.

**Retourne :**

un entier représentant la durée de perte de connection WWW tolérée (en secondes) avant de déclancher un redémarrage automatique pour tenter de récupérer la connectivité Internet

En cas d'erreur, déclenche une exception ou retourne `Y_WWWWATCHDOGDELAY_INVALID`.

## network→isOnline() YNetwork

Vérifie si le module hébergeant l'interface réseau est joignable, sans déclencher d'erreur.

```
js      function isOnline( )
nodejs  function isOnline( )
php     function isOnline( )
cpp     bool isOnline( )
m       -(BOOL) isOnline
pas     function isOnline( ): boolean
vb      function isOnline( ) As Boolean
cs      bool isOnline( )
java    boolean isOnline( )
py      def isOnline( )
```

Si les valeurs des attributs en cache de l'interface réseau sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

`true` si l'interface réseau est joignable, `false` sinon

## network→isOnline_async() YNetwork

Vérifie si le module hébergeant l'interface réseau est joignable, sans déclencher d'erreur.

```
js      function isOnline_async( callback, context)
nodejs  function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache de l'interface réseau sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

## network→load() YNetwork

Met en cache les valeurs courantes de l'interface réseau, avec une durée de validité spécifiée.

```
js      function load( msValidity)
nodejs  function load( msValidity)
php     function load( $msValidity)
cpp     YRETCODE load( int msValidity)
m       -(YRETCODE) load : (int) msValidity
pas     function load( msValidity: integer): YRETCODE
vb      function load( ByVal msValidity As Integer) As YRETCODE
cs      YRETCODE load( int msValidity)
java    int load( long msValidity)
py      def load( msValidity)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## network→load_async() YNetwork

Met en cache les valeurs courantes de l'interface réseau, avec une durée de validité spécifiée.

```
js      function load_async( msValidity, callback, context)
nodejs  function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou YAPI_SUCCESS)

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

## network→nextNetwork() YNetwork

Continue l'énumération des interfaces réseau commencée à l'aide de `yFirstNetwork()`.

```
js      function nextNetwork( )
nodejs  function nextNetwork( )
php     function nextNetwork( )
cpp     YNetwork * nextNetwork( )
m       -(YNetwork*) nextNetwork
pas     function nextNetwork( ): TYNetwork
vb      function nextNetwork( ) As YNetwork
cs      YNetwork nextNetwork( )
java    YNetwork nextNetwork( )
py      def nextNetwork( )
```

**Retourne :**

un pointeur sur un objet `YNetwork` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## network→ping() YNetwork

Ping str_host pour vérifier la connexion réseau.

```
js      function ping( host)
nodejs  function ping( host)
php     function ping( $host)
cpp     string ping( string host)
m       -(NSString*) ping : (NSString*) host
pas     function ping( host: string): string
vb      function ping( ) As String
cs      string ping( string host)
java    String ping( String host)
py      def ping( host)
cmd     YNetwork target ping host
```

Envoie quatre requêtes ICMP ECHO_RESPONSER à la cible str_host depuis le module. Cette méthode retourne une chaîne de caractères avec le résultat des 4 requêtes ICMP ECHO_RESPONSE.

**Paramètres :**

**host** le nom d'hôte ou l'adresse IP de la cible

**Retourne :**

une chaîne de caractères contenant le résultat du ping.

## network→registerValueCallback() YNetwork

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

```
js      function registerValueCallback( callback)
nodejs  function registerValueCallback( callback)
php     function registerValueCallback( $callback)
cpp     int registerValueCallback( YNetworkValueCallback callback)
m       -(int) registerValueCallback : (YNetworkValueCallback) callback
pas     function registerValueCallback( callback: TYNetworkValueCallback): LongInt
vb      function registerValueCallback( ) As Integer
cs      int registerValueCallback( ValueCallback callback)
java    int registerValueCallback( UpdateCallback callback)
py      def registerValueCallback( callback)
```

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

**Paramètres :**

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

## network→set_adminPassword() YNetwork
## network→setAdminPassword()

Modifie le mot de passe pour l'utilisateur "admin", qui devient alors instantanément nécessaire pour toute altération de l'état du module.

```
js      function set_adminPassword( newval)
nodejs  function set_adminPassword( newval)
php     function set_adminPassword( $newval)
cpp     int set_adminPassword( const string& newval)
m       -(int) setAdminPassword : (NSString*) newval
pas     function set_adminPassword( newval: string): integer
vb      function set_adminPassword( ByVal newval As String) As Integer
cs      int set_adminPassword( string newval)
java    int set_adminPassword( String newval)
py      def set_adminPassword( newval)
cmd     YNetwork target set_adminPassword newval
```

Si la valeur fournie est une chaîne vide, plus aucun mot de passe n'est nécessaire. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le mot de passe pour l'utilisateur "admin", qui devient alors instantanément nécessaire pour toute altération de l'état du module

**Retourne :**

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## network→set_callbackCredentials() YNetwork
## network→setCallbackCredentials()

Modifie le laisser-passer pour se connecter à l'adresse de callback.

```
js      function set_callbackCredentials( newval)
nodejs  function set_callbackCredentials( newval)
php     function set_callbackCredentials( $newval)
cpp     int set_callbackCredentials( const string& newval)
m       -(int) setCallbackCredentials : (NSString*) newval
pas     function set_callbackCredentials( newval: string): integer
vb      function set_callbackCredentials( ByVal newval As String) As Integer
cs      int set_callbackCredentials( string newval)
java    int set_callbackCredentials( String newval)
py      def set_callbackCredentials( newval)
cmd     YNetwork target set_callbackCredentials newval
```

Le laisser-passer doit être fourni tel que retourné par la fonction `get_callbackCredentials`, sous la forme `username:hash`. La valeur du hash dépend de la méthode d'autorisation implémentée par le callback. Pour une autorisation de type Basic, le hash est le MD5 de la chaîne `username:password`. Pour une autorisation de type Digest, le hash est le MD5 de la chaîne `username:realm:password`. Pour une utilisation simplifiée, utilisez la fonction `callbackLogin`. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le laisser-passer pour se connecter à l'adresse de callback

**Retourne :**

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## network→set_callbackEncoding() YNetwork
## network→setCallbackEncoding()

Modifie l'encodage à utiliser pour représenter les valeurs notifiées par callback.

```
js      function set_callbackEncoding( newval)
nodejs  function set_callbackEncoding( newval)
php     function set_callbackEncoding( $newval)
cpp     int set_callbackEncoding( Y_CALLBACKENCODING_enum newval)
m       -(int) setCallbackEncoding : (Y_CALLBACKENCODING_enum) newval
pas     function set_callbackEncoding( newval: Integer): integer
vb      function set_callbackEncoding( ByVal newval As Integer) As Integer
cs      int set_callbackEncoding( int newval)
java    int set_callbackEncoding( int newval)
py      def set_callbackEncoding( newval)
cmd     YNetwork target set_callbackEncoding newval
```

**Paramètres :**

**newval** une valeur parmi `Y_CALLBACKENCODING_FORM`, `Y_CALLBACKENCODING_JSON`, `Y_CALLBACKENCODING_JSON_ARRAY`, `Y_CALLBACKENCODING_CSV` et `Y_CALLBACKENCODING_YOCTO_API` représentant l'encodage à utiliser pour représenter les valeurs notifiées par callback

**Retourne :**

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## network→set_callbackMaxDelay() YNetwork
## network→setCallbackMaxDelay()

Modifie l'attente maximale entre deux notifications par callback, en secondes.

```
js      function set_callbackMaxDelay( newval)
nodejs  function set_callbackMaxDelay( newval)
php     function set_callbackMaxDelay( $newval)
cpp     int set_callbackMaxDelay( int newval)
m       -(int) setCallbackMaxDelay : (int) newval
pas     function set_callbackMaxDelay( newval: LongInt): integer
vb      function set_callbackMaxDelay( ByVal newval As Integer) As Integer
cs      int set_callbackMaxDelay( int newval)
java    int set_callbackMaxDelay( int newval)
py      def set_callbackMaxDelay( newval)
cmd     YNetwork target set_callbackMaxDelay newval
```

**Paramètres :**

**newval** un entier représentant l'attente maximale entre deux notifications par callback, en secondes

**Retourne :**

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## network→set_callbackMethod() YNetwork
## network→setCallbackMethod()

Modifie la méthode HTTP à utiliser pour signaler les changements d'état par callback.

```
js      function set_callbackMethod( newval)
nodejs  function set_callbackMethod( newval)
php     function set_callbackMethod( $newval)
cpp     int set_callbackMethod( Y_CALLBACKMETHOD_enum newval)
m       -(int) setCallbackMethod : (Y_CALLBACKMETHOD_enum) newval
pas     function set_callbackMethod( newval: Integer): integer
vb      function set_callbackMethod( ByVal newval As Integer) As Integer
cs      int set_callbackMethod( int newval)
java    int set_callbackMethod( int newval)
py      def set_callbackMethod( newval)
cmd     YNetwork target set_callbackMethod newval
```

**Paramètres :**

**newval** une valeur parmi `Y_CALLBACKMETHOD_POST`, `Y_CALLBACKMETHOD_GET` et `Y_CALLBACKMETHOD_PUT` représentant la méthode HTTP à utiliser pour signaler les changements d'état par callback

**Retourne :**

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## network→set_callbackMinDelay() YNetwork
## network→setCallbackMinDelay()

Modifie l'attente minimale entre deux notifications par callback, en secondes.

```
js      function set_callbackMinDelay( newval)
nodejs  function set_callbackMinDelay( newval)
php     function set_callbackMinDelay( $newval)
cpp     int set_callbackMinDelay( int newval)
m       -(int) setCallbackMinDelay : (int) newval
pas     function set_callbackMinDelay( newval: LongInt): integer
vb      function set_callbackMinDelay( ByVal newval As Integer) As Integer
cs      int set_callbackMinDelay( int newval)
java    int set_callbackMinDelay( int newval)
py      def set_callbackMinDelay( newval)
cmd     YNetwork target set_callbackMinDelay newval
```

**Paramètres :**

**newval** un entier représentant l'attente minimale entre deux notifications par callback, en secondes

**Retourne :**

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## network→set_callbackUrl() YNetwork
## network→setCallbackUrl()

Modifie l'adresse (URL) de callback à notifier lors de changement d'état significatifs.

```
js      function set_callbackUrl( newval)
nodejs  function set_callbackUrl( newval)
php     function set_callbackUrl( $newval)
cpp     int set_callbackUrl( const string& newval)
m       -(int) setCallbackUrl : (NSString*) newval
pas     function set_callbackUrl( newval: string): integer
vb      function set_callbackUrl( ByVal newval As String) As Integer
cs      int set_callbackUrl( string newval)
java    int set_callbackUrl( String newval)
py      def set_callbackUrl( newval)
cmd     YNetwork target set_callbackUrl newval
```

N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant l'adresse (URL) de callback à notifier lors de changement d'état significatifs

**Retourne :**

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## network→set_discoverable() YNetwork
## network→setDiscoverable()

Modifie l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocols uPnP/Bonjour).

```
js      function set_discoverable( newval)
nodejs  function set_discoverable( newval)
php     function set_discoverable( $newval)
cpp     int set_discoverable( Y_DISCOVERABLE_enum newval)
m       -(int) setDiscoverable : (Y_DISCOVERABLE_enum) newval
pas     function set_discoverable( newval: Integer): integer
vb      function set_discoverable( ByVal newval As Integer) As Integer
cs      int set_discoverable( int newval)
java    int set_discoverable( int newval)
py      def set_discoverable( newval)
cmd     YNetwork target set_discoverable newval
```

**Paramètres :**

**newval** soit `Y_DISCOVERABLE_FALSE`, soit `Y_DISCOVERABLE_TRUE`, selon l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocols uPnP/Bonjour)

**Retourne :**

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## network→set_logicalName() YNetwork
## network→setLogicalName()

Modifie le nom logique de l'interface réseau.

```
js      function set_logicalName( newval)
nodejs  function set_logicalName( newval)
php     function set_logicalName( $newval)
cpp     int set_logicalName( const string& newval)
m       -(int) setLogicalName : (NSString*) newval
pas     function set_logicalName( newval: string): integer
vb      function set_logicalName( ByVal newval As String) As Integer
cs      int set_logicalName( string newval)
java    int set_logicalName( String newval)
py      def set_logicalName( newval)
cmd     YNetwork target set_logicalName newval
```

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique de l'interface réseau.

**Retourne :**

`YAPI_SUCCESS` si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## network→set_primaryDNS() YNetwork
## network→setPrimaryDNS()

Modifie l'adresse IP du serveur de noms primaire que le module doit utiliser.

```
js      function set_primaryDNS( newval)
nodejs  function set_primaryDNS( newval)
php     function set_primaryDNS( $newval)
cpp     int set_primaryDNS( const string& newval)
m       -(int) setPrimaryDNS : (NSString*) newval
pas     function set_primaryDNS( newval: string): integer
vb      function set_primaryDNS( ByVal newval As String) As Integer
cs      int set_primaryDNS( string newval)
java    int set_primaryDNS( String newval)
py      def set_primaryDNS( newval)
cmd     YNetwork target set_primaryDNS newval
```

En mode DHCP, si une valeur est spécifiée, elle remplacera celle reçue du serveur DHCP. N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

**Paramètres :**

**newval** une chaîne de caractères représentant l'adresse IP du serveur de noms primaire que le module doit utiliser

**Retourne :**

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## network→set_secondaryDNS() YNetwork
## network→setSecondaryDNS()

Modifie l'adresse IP du serveur de nom secondaire que le module doit utiliser.

```
js      function set_secondaryDNS( newval)
nodejs  function set_secondaryDNS( newval)
php     function set_secondaryDNS( $newval)
cpp     int set_secondaryDNS( const string& newval)
m       -(int) setSecondaryDNS : (NSString*) newval
pas     function set_secondaryDNS( newval: string): integer
vb      function set_secondaryDNS( ByVal newval As String) As Integer
cs      int set_secondaryDNS( string newval)
java    int set_secondaryDNS( String newval)
py      def set_secondaryDNS( newval)
cmd     YNetwork target set_secondaryDNS newval
```

En mode DHCP, si une valeur est spécifiée, elle remplacera celle reçue du serveur DHCP. N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

**Paramètres :**

**newval** une chaîne de caractères représentant l'adresse IP du serveur de nom secondaire que le module doit utiliser

**Retourne :**

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## network→set_userData() YNetwork
## network→setUserData()

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

```
js      function set_userData( data)
nodejs  function set_userData( data)
php     function set_userData( $data)
cpp     void set_userData( void* data)
m       -(void) setUserData : (void*) data
pas     procedure set_userData( data: Tobject)
vb      procedure set_userData( ByVal data As Object)
cs      void set_userData( object data)
java    void set_userData( Object data)
py      def set_userData( data)
```

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

## network→set_userPassword() YNetwork
## network→setUserPassword()

Modifie le mode de passe pour l'utilisateur "user", qui devient alors instantanément nécessaire pour tout accès au module.

```
js      function set_userPassword( newval)
nodejs  function set_userPassword( newval)
php     function set_userPassword( $newval)
cpp     int set_userPassword( const string& newval)
m       -(int) setUserPassword : (NSString*) newval
pas     function set_userPassword( newval: string): integer
vb      function set_userPassword( ByVal newval As String) As Integer
cs      int set_userPassword( string newval)
java    int set_userPassword( String newval)
py      def set_userPassword( newval)
cmd     YNetwork target set_userPassword newval
```

Si la valeur fournie est une chaîne vide, plus aucun mot de passe n'est nécessaire. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le mode de passe pour l'utilisateur "user", qui devient alors instantanément nécessaire pour tout accès au module

**Retourne :**

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## network→set_wwwWatchdogDelay() YNetwork
## network→setWwwWatchdogDelay()

Modifie la durée de perte de connection WWW tolérée (en secondes) avant de déclancher un redémarrage automatique pour tenter de récupérer la connectivité Internet.

```
js      function set_wwwWatchdogDelay( newval)
nodejs  function set_wwwWatchdogDelay( newval)
php     function set_wwwWatchdogDelay( $newval)
cpp     int set_wwwWatchdogDelay( int newval)
m       -(int) setWwwWatchdogDelay : (int) newval
pas     function set_wwwWatchdogDelay( newval: LongInt): integer
vb      function set_wwwWatchdogDelay( ByVal newval As Integer) As Integer
cs      int set_wwwWatchdogDelay( int newval)
java    int set_wwwWatchdogDelay( int newval)
py      def set_wwwWatchdogDelay( newval)
cmd     YNetwork target set_wwwWatchdogDelay newval
```

Une valeur nulle désactive le redémarrage automatique en cas de perte de connectivité WWW. La plus petite durée non-nulle utilisable est 90 secondes.

**Paramètres :**

**newval** un entier représentant la durée de perte de connection WWW tolérée (en secondes) avant de déclancher un redémarrage automatique pour tenter de récupérer la connectivité Internet

**Retourne :**

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## network→useDHCP() YNetwork

Modifie la configuration de l'interface réseau pour utiliser une adresse assignée automatiquement par le serveur DHCP.

```
js      function useDHCP( fallbackIpAddr, fallbackSubnetMaskLen, fallbackRouter)
nodejs  function useDHCP( fallbackIpAddr, fallbackSubnetMaskLen, fallbackRouter)
php     function useDHCP( $fallbackIpAddr, $fallbackSubnetMaskLen, $fallbackRouter)
cpp     int useDHCP( string fallbackIpAddr,
                     int fallbackSubnetMaskLen,
                     string fallbackRouter)
m       -(int) useDHCP : (NSString*) fallbackIpAddr
                       : (int) fallbackSubnetMaskLen
                       : (NSString*) fallbackRouter
pas     function useDHCP( fallbackIpAddr: string,
                          fallbackSubnetMaskLen: LongInt,
                          fallbackRouter: string): integer
vb      function useDHCP( ByVal fallbackIpAddr As String,
                          ByVal fallbackSubnetMaskLen As Integer,
                          ByVal fallbackRouter As String) As Integer
cs      int useDHCP( string fallbackIpAddr,
                     int fallbackSubnetMaskLen,
                     string fallbackRouter)
java    int useDHCP( String fallbackIpAddr,
                     int fallbackSubnetMaskLen,
                     String fallbackRouter)
py      def useDHCP( fallbackIpAddr, fallbackSubnetMaskLen, fallbackRouter)
cmd     YNetwork target useDHCP fallbackIpAddr fallbackSubnetMaskLen fallbackRouter
```

En attendant qu'une adresse soit reçue (et indéfiniment si aucun serveur DHCP ne répond), le module utilisera les paramètres IP spécifiés à cette fonction. N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

**Paramètres :**

| | |
|---|---|
| **fallbackIpAddr** | adresse IP à utiliser si aucun serveur DHCP ne répond |
| **fallbackSubnetMaskLen** | longueur du masque de sous-réseau à utiliser si aucun serveur DHCP ne répond. Par exemple, la valeur 24 représente 255.255.255.0. |
| **fallbackRouter** | adresse de la passerelle à utiliser si aucun serveur DHCP ne répond |

**Retourne :**

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## network→useStaticIP() YNetwork

Modifie la configuration de l'interface réseau pour utiliser une adresse IP assignée manuellement (adresse IP statique).

```
js      function useStaticIP( ipAddress, subnetMaskLen, router)
nodejs  function useStaticIP( ipAddress, subnetMaskLen, router)
php     function useStaticIP( $ipAddress, $subnetMaskLen, $router)
cpp     int useStaticIP( string ipAddress,
                         int subnetMaskLen,
                         string router)
m       -(int) useStaticIP : (NSString*) ipAddress
                           : (int) subnetMaskLen
                           : (NSString*) router
pas     function useStaticIP( ipAddress: string,
                              subnetMaskLen: LongInt,
                              router: string): integer
vb      function useStaticIP( ByVal ipAddress As String,
                              ByVal subnetMaskLen As Integer,
                              ByVal router As String) As Integer
cs      int useStaticIP( string ipAddress,
                         int subnetMaskLen,
                         string router)
java    int useStaticIP( String ipAddress,
                         int subnetMaskLen,
                         String router)
py      def useStaticIP( ipAddress, subnetMaskLen, router)
cmd     YNetwork target useStaticIP ipAddress subnetMaskLen router
```

N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

**Paramètres :**

| | |
|---|---|
| **ipAddress** | adresse IP à utiliser par le module |
| **subnetMaskLen** | longueur du masque de sous-réseau à utiliser. Par exemple, la valeur 24 représente 255.255.255.0. |
| **router** | adresse IP de la passerelle à utiliser ("default gateway") |

**Retourne :**

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## network→wait_async() YNetwork

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js      function wait_async( callback, context)
nodejs  function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

# 10.3. Interface de la fonction Files

L'interface de stockage de fichiers permet de stocker des fichiers sur certains modules, par exemple pour personnaliser un service web (dans le cas d'un module connecté au réseau) ou pour ajouter un police de caractères (dans le cas d'un module d'affichage).

Pour utiliser les fonctions décrites ici, vous devez inclure:

```
js      <script type='text/javascript' src='yocto_files.js'></script>
nodejs  var yoctolib = require('yoctolib');
        var YFiles = yoctolib.YFiles;
php     require_once('yocto_files.php');
cpp     #include "yocto_files.h"
m       #import "yocto_files.h"
pas     uses yocto_files;
vb      yocto_files.vb
cs      yocto_files.cs
java    import com.yoctopuce.YoctoAPI.YFiles;
py      from yocto_files import *
```

| Fonction globales |
|---|
| **yFindFiles**(**func**)<br>Permet de retrouver un système de fichier d'après un identifiant donné. |
| **yFirstFiles**()<br>Commence l'énumération des système de fichier accessibles par la librairie. |
| **Méthodes des objets `YFiles`** |
| **files→describe**()<br>Retourne un court texte décrivant le système de fichier au format `TYPE(NAME)=SERIAL.FUNCTIONID`. |
| **files→download**(**pathname**)<br>Télécharge le fichier choisi du filesystème et retourne son contenu. |
| **files→download_async**(**pathname**, **callback**, **context**)<br>Procède au chargement du bloc suivant de mesures depuis l'enregistreur de données du module, de manière asynchrone. |
| **files→format_fs**()<br>Rétabli le système de fichier dans on état original, défragmenté. |
| **files→get_advertisedValue**()<br>Retourne la valeur courante du système de fichier (pas plus de 6 caractères). |
| **files→get_errorMessage**()<br>Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du système de fichier. |
| **files→get_errorType**()<br>Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du système de fichier. |
| **files→get_filesCount**()<br>Retourne le nombre de fichiers présents dans le système de fichier. |
| **files→get_freeSpace**()<br>Retourne l'espace disponible dans le système de fichier pour charger des nouveaux fichiers, en octets. |
| **files→get_friendlyName**()<br>Retourne un identifiant global du système de fichier au format `NOM_MODULE.NOM_FONCTION`. |
| **files→get_functionDescriptor**() |

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

**files→get_functionId**()

Retourne l'identifiant matériel du système de fichier, sans référence au module.

**files→get_hardwareId**()

Retourne l'identifiant matériel unique du système de fichier au format SERIAL.FUNCTIONID.

**files→get_list**(**pattern**)

Retourne une liste d'objets objet YFileRecord qui décrivent les fichiers présents dans le système de fichier.

**files→get_logicalName**()

Retourne le nom logique du système de fichier.

**files→get_module**()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

**files→get_module_async**(**callback**, **context**)

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

**files→get_userData**()

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set_userData.

**files→isOnline**()

Vérifie si le module hébergeant le système de fichier est joignable, sans déclencher d'erreur.

**files→isOnline_async**(**callback**, **context**)

Vérifie si le module hébergeant le système de fichier est joignable, sans déclencher d'erreur.

**files→load**(**msValidity**)

Met en cache les valeurs courantes du système de fichier, avec une durée de validité spécifiée.

**files→load_async**(**msValidity**, **callback**, **context**)

Met en cache les valeurs courantes du système de fichier, avec une durée de validité spécifiée.

**files→nextFiles**()

Continue l'énumération des système de fichier commencée à l'aide de yFirstFiles().

**files→registerValueCallback**(**callback**)

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

**files→remove**(**pathname**)

Efface un fichier, spécifié par son path complet, du système de fichier.

**files→set_logicalName**(**newval**)

Modifie le nom logique du système de fichier.

**files→set_userData**(**data**)

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode get_userData.

**files→upload**(**pathname**, **content**)

Télécharge un contenu vers le système de fichier, au chemin d'accès spécifié.

**files→wait_async**(**callback**, **context**)

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YFiles.FindFiles() YFiles
## yFindFiles()

Permet de retrouver un système de fichier d'après un identifiant donné.

```
js      function yFindFiles( func)
nodejs  function FindFiles( func)
php     function yFindFiles( $func)
cpp     YFiles* yFindFiles( string func)
m       +(YFiles*) yFindFiles : (NSString*) func
pas     function yFindFiles( func: string): TYFiles
vb      function yFindFiles( ByVal func As String) As YFiles
cs      YFiles FindFiles( string func)
java    YFiles FindFiles( String func)
py      def FindFiles( func)
```

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le système de fichier soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YFiles.isOnline()` pour tester si le système de fichier est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

**Paramètres :**

**func** une chaîne de caractères qui référence le système de fichier sans ambiguïté

**Retourne :**

un objet de classe `YFiles` qui permet ensuite de contrôler le système de fichier.

## YFiles.FirstFiles() YFiles
## yFirstFiles()

Commence l'énumération des système de fichier accessibles par la librairie.

```
js      function yFirstFiles( )
nodejs  function FirstFiles( )
php     function yFirstFiles( )
cpp     YFiles* yFirstFiles( )
m       YFiles* yFirstFiles( )
pas     function yFirstFiles( ): TYFiles
vb      function yFirstFiles( ) As YFiles
cs      YFiles FirstFiles( )
java    YFiles FirstFiles( )
py      def FirstFiles( )
```

Utiliser la fonction `YFiles.nextFiles()` pour itérer sur les autres système de fichier.

**Retourne :**

un pointeur sur un objet `YFiles`, correspondant à le premier système de fichier accessible en ligne, ou `null` si il n'y a pas de système de fichier disponibles.

## files→describe() YFiles

Retourne un court texte décrivant le système de fichier au format TYPE(NAME)=SERIAL.FUNCTIONID.

```
js      function describe( )
nodejs  function describe( )
php     function describe( )
cpp     string describe( )
m       -(NSString*) describe
pas     function describe( ): string
vb      function describe( ) As String
cs      string describe( )
java    String describe( )
py      def describe( )
```

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilsé lors du premier accès a la fonction, SERIAL correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et FUNCTIONID correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La methode va retourner Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1 si le module est déjà connecté ou Relay(BadCustomeName.relay1)=unresolved si le module n'est pas déjà connecté. Cette methode ne declenche aucune transaction USB ou TCP et peut donc être utilisé dans un debuggeur.

**Retourne :**
une chaîne de caractères décrivant le système de fichier (ex: Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1)

## files→download() YFiles

Télécharge le fichier choisi du filesystème et retourne son contenu.

```
js      function download( pathname)
nodejs  function download( pathname)
php     function download( $pathname)
cpp     string download( string pathname)
m       -(NSData*) download : (NSString*) pathname
pas     function download( pathname: string): TByteArray
vb      function download( ) As Byte
py      def download( pathname)
cmd     YFiles target download pathname
```

**Paramètres :**

**pathname** nom complet du fichier à charger, y compris le chemin d'accès.

**Retourne :**

le contenu du fichier chargé sous forme d'objet binaire

En cas d'erreur, déclenche une exception ou retourne un contenu vide.

## files→download_async() YFiles

Procède au chargement du bloc suivant de mesures depuis l'enregistreur de données du module, de manière asynchrone.

```
js      function download_async( pathname, callback, context)
nodejs  function download_async( pathname, callback, context)
```

**Paramètres :**

**pathname** nom complet du fichier à charger, y compris le chemin d'accès.

**callback** fonction fournie par l'utilisateur, qui sera appelée lorsque la suite du chargement aura été effectué. La fonction callback doit prendre trois arguments: - la variable de contexte à disposition de l'utilisateur - l'objet YFiles dont la méthode download_async a été appelée - le contenu du fichier chargé sous forme d'objet binaire

**context** variable de contexte à disposition de l'utilisateur

**Retourne :**

rien.

## files→format_fs() YFiles

Rétabli le système de fichier dans on état original, défragmenté.

```
js      function format_fs( )
nodejs  function format_fs( )
php     function format_fs( )
cpp     int format_fs( )
m       -(int) format_fs
pas     function format_fs( ): LongInt
vb      function format_fs( ) As Integer
cs      int format_fs( )
java    int format_fs( )
py      def format_fs( )
cmd     YFiles target format_fs
```

entièrement vide. Tous les fichiers précédemment chargés sont irrémédiablement effacés.

**Retourne :**

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## files→get_advertisedValue() YFiles
## files→advertisedValue()

Retourne la valeur courante du système de fichier (pas plus de 6 caractères).

```
js      function get_advertisedValue( )
nodejs  function get_advertisedValue( )
php     function get_advertisedValue( )
cpp     string get_advertisedValue( )
m       -(NSString*) advertisedValue
pas     function get_advertisedValue( ): string
vb      function get_advertisedValue( ) As String
cs      string get_advertisedValue( )
java    String get_advertisedValue( )
py      def get_advertisedValue( )
cmd     YFiles target get_advertisedValue
```

**Retourne :**

une chaîne de caractères représentant la valeur courante du système de fichier (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne `Y_ADVERTISEDVALUE_INVALID`.

## files→get_errorMessage() YFiles
## files→errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du système de fichier.

```
js       function get_errorMessage( )
nodejs   function get_errorMessage( )
php      function get_errorMessage( )
cpp      string get_errorMessage( )
m        -(NSString*) errorMessage
pas      function get_errorMessage( ): string
vb       function get_errorMessage( ) As String
cs       string get_errorMessage( )
java     String get_errorMessage( )
py       def get_errorMessage( )
```

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du système de fichier.

## files→get_errorType() YFiles
## files→errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du système de fichier.

```
js      function get_errorType( )
nodejs  function get_errorType( )
php     function get_errorType( )
cpp     YRETCODE get_errorType( )
pas     function get_errorType( ): YRETCODE
vb      function get_errorType( ) As YRETCODE
cs      YRETCODE get_errorType( )
java    int get_errorType( )
py      def get_errorType( )
```

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du système de fichier.

## files→get_filesCount() YFiles
## files→filesCount()

Retourne le nombre de fichiers présents dans le système de fichier.

```
js      function get_filesCount( )
nodejs  function get_filesCount( )
php     function get_filesCount( )
cpp     int get_filesCount( )
m       -(int) filesCount
pas     function get_filesCount( ): LongInt
vb      function get_filesCount( ) As Integer
cs      int get_filesCount( )
java    int get_filesCount( )
py      def get_filesCount( )
cmd     YFiles target get_filesCount
```

**Retourne :**
un entier représentant le nombre de fichiers présents dans le système de fichier

En cas d'erreur, déclenche une exception ou retourne `Y_FILESCOUNT_INVALID`.

## files→get_freeSpace() YFiles
## files→freeSpace()

Retourne l'espace disponible dans le système de fichier pour charger des nouveaux fichiers, en octets.

```
js      function get_freeSpace( )
nodejs  function get_freeSpace( )
php     function get_freeSpace( )
cpp     int get_freeSpace( )
m       -(int) freeSpace
pas     function get_freeSpace( ): LongInt
vb      function get_freeSpace( ) As Integer
cs      int get_freeSpace( )
java    int get_freeSpace( )
py      def get_freeSpace( )
cmd     YFiles target get_freeSpace
```

**Retourne :**

un entier représentant l'espace disponible dans le système de fichier pour charger des nouveaux fichiers, en octets

En cas d'erreur, déclenche une exception ou retourne `Y_FREESPACE_INVALID`.

## files→get_friendlyName() YFiles
## files→friendlyName()

Retourne un identifiant global du système de fichier au format NOM_MODULE.NOM_FONCTION.

```
js      function get_friendlyName( )
nodejs  function get_friendlyName( )
php     function get_friendlyName( )
cpp     string get_friendlyName( )
m       -(NSString*) friendlyName
cs      string get_friendlyName( )
java    String get_friendlyName( )
py      def get_friendlyName( )
```

Le chaîne retournée utilise soit les noms logiques du module et du système de fichier si ils sont définis, soit respectivement le numéro de série du module et l'identifant matériel du système de fichier (par exemple: MyCustomName.relay1)

**Retourne :**

une chaîne de caractères identifiant le système de fichier en utilisant les noms logiques (ex: MyCustomName.relay1) En cas d'erreur, déclenche une exception ou retourne Y_FRIENDLYNAME_INVALID.

## files→get_functionDescriptor() YFiles
## files→functionDescriptor()

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

```
js       function get_functionDescriptor( )
nodejs   function get_functionDescriptor( )
php      function get_functionDescriptor( )
cpp      YFUN_DESCR get_functionDescriptor( )
m        -(YFUN_DESCR) functionDescriptor
pas      function get_functionDescriptor( ): YFUN_DESCR
vb       function get_functionDescriptor( ) As YFUN_DESCR
cs       YFUN_DESCR get_functionDescriptor( )
java     String get_functionDescriptor( )
py       def get_functionDescriptor( )
```

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y_FUNCTIONDESCRIPTOR_INVALID

## files→get_functionId() YFiles
## files→functionId()

Retourne l'identifiant matériel du système de fichier, sans référence au module.

```
js      function get_functionId( )
nodejs  function get_functionId( )
php     function get_functionId( )
cpp     string get_functionId( )
m       -(NSString*) functionId
vb      function get_functionId( ) As String
cs      string get_functionId( )
java    String get_functionId( )
py      def get_functionId( )
```

Par example `relay1`.

**Retourne :**

une chaîne de caractères identifiant le système de fichier (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

## files→get_hardwareId() YFiles
## files→hardwareId()

Retourne l'identifiant matériel unique du système de fichier au format SERIAL.FUNCTIONID.

```
js      function get_hardwareId( )
nodejs  function get_hardwareId( )
php     function get_hardwareId( )
cpp     string get_hardwareId( )
m       -(NSString*) hardwareId
vb      function get_hardwareId( ) As String
cs      string get_hardwareId( )
java    String get_hardwareId( )
py      def get_hardwareId( )
```

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du système de fichier (par example RELAYLO1-123456.relay1).

**Retourne :**

une chaîne de caractères identifiant le système de fichier (ex: RELAYLO1-123456.relay1) En cas d'erreur, déclenche une exception ou retourne Y_HARDWAREID_INVALID.

## files→get_list() YFiles
## files→list()

Retourne une liste d'objets objet YFileRecord qui décrivent les fichiers présents dans le système de fichier.

```
js      function get_list( pattern)
nodejs  function get_list( pattern)
php     function get_list( $pattern)
cpp     vector<YFileRecord> get_list( string pattern)
m       -(NSMutableArray*) list : (NSString*) pattern
pas     function get_list( pattern: string): TYFileRecordArray
vb      function get_list( ) As List
cs      List<YFileRecord> get_list( string pattern)
java    ArrayList<YFileRecord> get_list( String pattern)
py      def get_list( pattern)
cmd     YFiles target get_list pattern
```

**Paramètres :**

**pattern** un filtre optionel sur les noms de fichiers retournés, pouvant contenir des astérisques et des points d'interrogations comme jokers. Si le pattern fourni est vide, tous les fichiers sont retournés.

**Retourne :**

une liste d'objets `YFileRecord`, contenant le nom complet (y compris le chemin d'accès), la taille en octets et le CRC 32-bit du contenu du fichier.

En cas d'erreur, déclenche une exception ou retourne une liste vide.

## files→get_logicalName() YFiles
## files→logicalName()

Retourne le nom logique du système de fichier.

```
js      function get_logicalName( )
nodejs  function get_logicalName( )
php     function get_logicalName( )
cpp     string get_logicalName( )
m       -(NSString*) logicalName
pas     function get_logicalName( ): string
vb      function get_logicalName( ) As String
cs      string get_logicalName( )
java    String get_logicalName( )
py      def get_logicalName( )
cmd     YFiles target get_logicalName
```

**Retourne :**

une chaîne de caractères représentant le nom logique du système de fichier. En cas d'erreur, déclenche une exception ou retourne `Y_LOGICALNAME_INVALID`.

## files→get_module() YFiles
## files→module()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

```
js      function get_module( )
nodejs  function get_module( )
php     function get_module( )
cpp     YModule * get_module( )
m       -(YModule*) module
pas     function get_module( ): TYModule
vb      function get_module( ) As YModule
cs      YModule get_module( )
java    YModule get_module( )
py      def get_module( )
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

**Retourne :**
une instance de YModule

## files→get_module_async() YFiles
## files→module_async()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

```
js      function get_module_async( callback, context)
nodejs  function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de YModule

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

## files→get_userData() YFiles
## files→userData()

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

```
js      function get_userData( )
nodejs  function get_userData( )
php     function get_userData( )
cpp     void * get_userData( )
m       -(void*) userData
pas     function get_userData( ): Tobject
vb      function get_userData( ) As Object
cs      object get_userData( )
java    Object get_userData( )
py      def get_userData( )
```

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**
l'objet stocké précédemment par l'appelant.

## files→isOnline() YFiles

Vérifie si le module hébergeant le système de fichier est joignable, sans déclencher d'erreur.

```
js      function isOnline( )
nodejs  function isOnline( )
php     function isOnline( )
cpp     bool isOnline( )
m       -(BOOL) isOnline
pas     function isOnline( ): boolean
vb      function isOnline( ) As Boolean
cs      bool isOnline( )
java    boolean isOnline( )
py      def isOnline( )
```

Si les valeurs des attributs en cache du système de fichier sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

`true` si le système de fichier est joignable, `false` sinon

## files→isOnline_async() YFiles

Vérifie si le module hébergeant le système de fichier est joignable, sans déclencher d'erreur.

```
js      function isOnline_async( callback, context)
nodejs  function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du système de fichier sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

## files→load() YFiles

Met en cache les valeurs courantes du système de fichier, avec une durée de validité spécifiée.

```
js      function load( msValidity)
nodejs  function load( msValidity)
php     function load( $msValidity)
cpp     YRETCODE load( int msValidity)
m       -(YRETCODE) load : (int) msValidity
pas     function load( msValidity: integer): YRETCODE
vb      function load( ByVal msValidity As Integer) As YRETCODE
cs      YRETCODE load( int msValidity)
java    int load( long msValidity)
py      def load( msValidity)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## files→load_async() YFiles

Met en cache les valeurs courantes du système de fichier, avec une durée de validité spécifiée.

```
js      function load_async( msValidity, callback, context)
nodejs  function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou YAPI_SUCCESS)

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

## files→nextFiles() YFiles

Continue l'énumération des système de fichier commencée à l'aide de `yFirstFiles()`.

```
js      function nextFiles( )
nodejs  function nextFiles( )
php     function nextFiles( )
cpp     YFiles * nextFiles( )
m       -(YFiles*) nextFiles
pas     function nextFiles( ): TYFiles
vb      function nextFiles( ) As YFiles
cs      YFiles nextFiles( )
java    YFiles nextFiles( )
py      def nextFiles( )
```

**Retourne :**

un pointeur sur un objet `YFiles` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## files→registerValueCallback() YFiles

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

```
js      function registerValueCallback( callback)
nodejs  function registerValueCallback( callback)
php     function registerValueCallback( $callback)
cpp     int registerValueCallback( YFilesValueCallback callback)
m       -(int) registerValueCallback : (YFilesValueCallback) callback
pas     function registerValueCallback( callback: TYFilesValueCallback): LongInt
vb      function registerValueCallback( ) As Integer
cs      int registerValueCallback( ValueCallback callback)
java    int registerValueCallback( UpdateCallback callback)
py      def registerValueCallback( callback)
```

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

**Paramètres :**

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

## files→remove() YFiles

Efface un fichier, spécifié par son path complet, du système de fichier.

```
js      function remove( pathname)
nodejs  function remove( pathname)
php     function remove( $pathname)
cpp     int remove( string pathname)
m       -(int) remove : (NSString*) pathname
pas     function remove( pathname: string): LongInt
vb      function remove( ) As Integer
cs      int remove( string pathname)
java    int remove( String pathname)
py      def remove( pathname)
cmd     YFiles target remove pathname
```

A cause de la fragmentation, l'effacement d'un fichier ne libère pas toujours la totalité de l'espace qu'il occuppe. Par contre, la ré-écriture d'un fichier du même nom récupérera dans tout les cas l'espace qui n'aurait éventuellement pas été libéré. Pour s'assurer de libérer la totalité de l'espace du système de fichier, utilisez la fonction `format_fs`.

**Paramètres :**

**pathname** nom complet du fichier, y compris le chemin d'accès.

**Retourne :**

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## files→set_logicalName() YFiles
## files→setLogicalName()

Modifie le nom logique du système de fichier.

```
js      function set_logicalName( newval)
nodejs  function set_logicalName( newval)
php     function set_logicalName( $newval)
cpp     int set_logicalName( const string& newval)
m       -(int) setLogicalName : (NSString*) newval
pas     function set_logicalName( newval: string): integer
vb      function set_logicalName( ByVal newval As String) As Integer
cs      int set_logicalName( string newval)
java    int set_logicalName( String newval)
py      def set_logicalName( newval)
cmd     YFiles target set_logicalName newval
```

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique du système de fichier.

**Retourne :**

`YAPI_SUCCESS` si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## files→set_userData() YFiles
## files→setUserData()

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

```
js      function set_userData( data)
nodejs  function set_userData( data)
php     function set_userData( $data)
cpp     void set_userData( void* data)
m       -(void) setUserData : (void*) data
pas     procedure set_userData( data: Tobject)
vb      procedure set_userData( ByVal data As Object)
cs      void set_userData( object data)
java    void set_userData( Object data)
py      def set_userData( data)
```

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

## files→upload() YFiles

Télécharge un contenu vers le système de fichier, au chemin d'accès spécifié.

```
js      function upload( pathname, content)
nodejs  function upload( pathname, content)
php     function upload( $pathname, $content)
cpp     int upload( string pathname, string content)
m       -(int) upload : (NSString*) pathname
                      : (NSData*) content
pas     function upload( pathname: string, content: TByteArray): LongInt
vb      procedure upload( )
cs      int upload( string pathname)
java    int upload( String pathname)
py      def upload( pathname, content)
cmd     YFiles target upload pathname content
```

Si un fichier existe déjà pour le même chemin d'accès, son contenu est remplacé.

**Paramètres :**

**pathname** nom complet du fichier, y compris le chemin d'accès.

**content** contenu du fichier à télécharger

**Retourne :**

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## files→wait_async() YFiles

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js      function wait_async( callback, context)
nodejs  function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

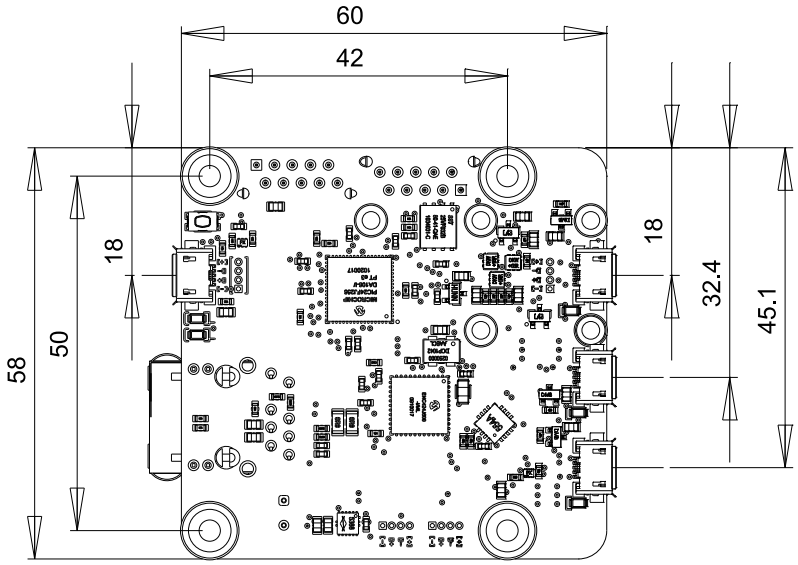
**Retourne :**

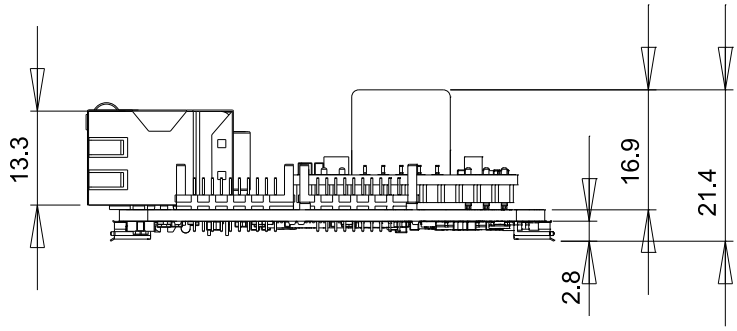
rien du tout :

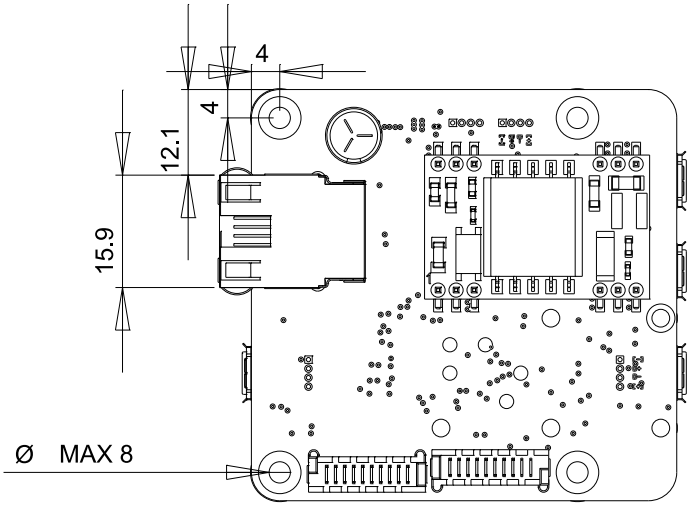
# 11. Caractéristiques

Vous trouverez résumées ci dessous les principales caractéristiques techniques de votre module YoctoHub-Ethernet

| | |
|---|---|
| Epaisseur | 22 mm |
| Largeur | 58 mm |
| Longueur | 60 mm |
| Poids | 34 g |
| Connecteur USB | micro-B |
| Canaux | 3 ports |
| Courant Max (continu) | 2 A |
| Consommation USB | 130 mA |
| Sytème d'exploitation supportés | Windows, Linux (Intel + ARM), Mac OS X, Android |
| Connection réseau | Fast-Ethernet |
| Drivers | Fonctionne sans driver |
| API / SDK / Librairie (USB+TCP) | C++, Objective-C, C#, VB .NET, Delphi, Python, Java/Android |
| API / SDK / Librairie (seul.TCP) | Javascript, Node.js, PHP, Java |
| RoHS | oui |
| USB Vendor ID | 0x24E0 |
| USB Device ID | 0x000E |
| Boîter recommandé | YoctoBox-HubEth-Transp |




All dimensions are in mm
Toutes les dimensions sont en mm

# YoctoHub-Ethernet

A4

Scale
1:1
Echelle

# Index


## A
Accès 21, 22, 27
Admin 22
Automatisée 10

## B
Blueprint 157

## C
Callback 24, 26
callbackLogin, YNetwork 66
Caractéristiques 155
Configuration 8, 10, 18, 23
Connectés 27
Connexions 11
Contrôle 21, 27

## D
Defined 24
describe, YFiles 125
describe, YHubPort 35
describe, YNetwork 67
Directe 7
download, YFiles 126
download_async, YFiles 127

## E
Effacement 22
Éléments 3
Extérieur 23

## F
Files 122
FindFiles, YFiles 123
FindHubPort, YHubPort 33
FindNetwork, YNetwork 64
Firmwares 19
FirstFiles, YFiles 124
FirstHubPort, YHubPort 34
FirstNetwork, YNetwork 65
Fixation 15, 16
format_fs, YFiles 128

## G
get_adminPassword, YNetwork 68
get_advertisedValue, YFiles 129
get_advertisedValue, YHubPort 36
get_advertisedValue, YNetwork 69
get_baudRate, YHubPort 37
get_callbackCredentials, YNetwork 70
get_callbackEncoding, YNetwork 71
get_callbackMaxDelay, YNetwork 72
get_callbackMethod, YNetwork 73
get_callbackMinDelay, YNetwork 74
get_callbackUrl, YNetwork 75
get_discoverable, YNetwork 76
get_enabled, YHubPort 38
get_errorMessage, YFiles 130
get_errorMessage, YHubPort 39
get_errorMessage, YNetwork 77
get_errorType, YFiles 131
get_errorType, YHubPort 40
get_errorType, YNetwork 78
get_filesCount, YFiles 132
get_freeSpace, YFiles 133
get_friendlyName, YFiles 134
get_friendlyName, YHubPort 41
get_friendlyName, YNetwork 79
get_functionDescriptor, YFiles 135
get_functionDescriptor, YHubPort 42
get_functionDescriptor, YNetwork 80
get_functionId, YFiles 136
get_functionId, YHubPort 43
get_functionId, YNetwork 81
get_hardwareId, YFiles 137
get_hardwareId, YHubPort 44
get_hardwareId, YNetwork 82
get_ipAddress, YNetwork 83
get_list, YFiles 138
get_logicalName, YFiles 139
get_logicalName, YHubPort 45
get_logicalName, YNetwork 84
get_macAddress, YNetwork 85
get_module, YFiles 140
get_module, YHubPort 46
get_module, YNetwork 86
get_module_async, YFiles 141
get_module_async, YHubPort 47
get_module_async, YNetwork 87
get_poeCurrent, YNetwork 88
get_portState, YHubPort 48
get_primaryDNS, YNetwork 89
get_readiness, YNetwork 90
get_router, YNetwork 91
get_secondaryDNS, YNetwork 92
get_subnetMask, YNetwork 93
get_userData, YFiles 142
get_userData, YHubPort 49
get_userData, YNetwork 94
get_userPassword, YNetwork 95
get_wwwWatchdogDelay, YNetwork 96

## I
Impatients 7
Influence 22

Interactions 23
Interface 29, 32, 61, 122
Introduction 1
isOnline, YFiles 143
isOnline, YHubPort 50
isOnline, YNetwork 97
isOnline_async, YFiles 144
isOnline_async, YHubPort 51
isOnline_async, YNetwork 98

# L

Limitations 30
load, YFiles 145
load, YHubPort 52
load, YNetwork 99
load_async, YFiles 146
load_async, YHubPort 53
load_async, YNetwork 100
Localisation 17

# M

Manuelle 8
Modules 17, 18, 27
Montage 15
Mots 22

# N

Network 61
nextFiles, YFiles 147
nextHubPort, YHubPort 54
nextNetwork, YNetwork 101
Niveau 31

# P

Passe 22
Personnalisation 29
ping, YNetwork 102
Port 32
Pour 7
Présentation 3
Programmation 27
Protégé 22

# R

Référence 31
registerValueCallback, YFiles 148
registerValueCallback, YHubPort 55
registerValueCallback, YNetwork 103
remove, YFiles 149

# S

set_adminPassword, YNetwork 104
set_callbackCredentials, YNetwork 105
set_callbackEncoding, YNetwork 106
set_callbackMaxDelay, YNetwork 107
set_callbackMethod, YNetwork 108
set_callbackMinDelay, YNetwork 109
set_callbackUrl, YNetwork 110
set_discoverable, YNetwork 111
set_enabled, YHubPort 56
set_logicalName, YFiles 150
set_logicalName, YHubPort 57
set_logicalName, YNetwork 112
set_primaryDNS, YNetwork 113
set_secondaryDNS, YNetwork 114
set_userData, YFiles 151
set_userData, YHubPort 58
set_userData, YNetwork 115
set_userPassword, YNetwork 116
set_wwwWatchdogDelay, YNetwork 117
Sous-module 16

# T

Test 18
Thinkspeak 26

# U

Upgrades 19
upload, YFiles 152
useDHCP, YNetwork 118
User 22, 24
useStaticIP, YNetwork 119

# W

wait_async, YFiles 153
wait_async, YHubPort 59
wait_async, YNetwork 120

# Y

YFiles 123-153
yFindFiles 123
yFindHubPort 33
yFindNetwork 64
yFirstFiles 124
yFirstHubPort 34
yFirstNetwork 65
YHubPort 33-59
YNetwork 64-120
Yocto-API 26
Yocto-hub 32
YoctoHub-Ethernet 3, 17, 27